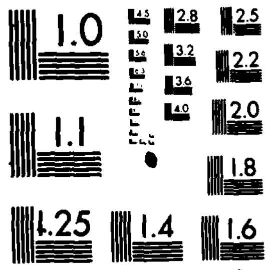
AD-A115 012 ARMY MATERIEL SYSTEMS ANALYSIS ACTIVITY ABERDEEN PROV--ETC F/G 9/2
GPSS AND MODELING OF COMPUTER COMMUNICATION NETWORKS.(U)
APR 82 C B SILIO, K F SCHWARZ
UNCLASSIFIED AMSAA-TR-352 NL

1 of 2

1 OF 2

AD
A115012



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD A115012

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AMSAA Technical Report No. 352	2. GOVT ACCESSION NO. AD-A115012	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) GPSS and Modeling of Computer Communication Networks		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Charles B. Silio, Jr. Kurt F. Schwarz		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS US Army Materiel Systems Analysis Activity Aberdeen Proving Ground, MD 21005		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS DA Project No. 1R765706M541
11. CONTROLLING OFFICE NAME AND ADDRESS US Army Materiel Development and Readiness Command 5001 Eisenhower Avenue Alexandria, VA 22333		12. REPORT DATE April 1982
		13. NUMBER OF PAGES 112
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

GPSS
Discrete Event Simulation
Modeling
Ring Networks

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The suitability of the language GPSS for discrete event simulation of computer communication networks is investigated. Capabilities and peculiarities of GPSS are examined, and sample simulation results for four ring network link level protocol models are presented. Translation of models from one dialect of GPSS to another is discussed, and certain inherent semantic differences are identified.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ACKNOWLEDGEMENTS

We thank Dr. A. Brinton Cooper III, Patrick J. Ward and Ruth R. Stanley of the C^{3}A Branch, CSD, AMSAA, for their valued support and assistance throughout this project. The assistance of Mr. Ward and Ms. Stanley was particularly helpful in identifying semantic differences in the dialects of GPSS. We also thank Professor M. T. Liu of the Ohio State University for kindly supplying listings of GPSS simulation programs generated by two of his former students.

Next page is blank.

CONTENTS

CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

Next page is blank.

GPSS AND MODELING OF COMPUTER COMMUNICATION NETWORKS

1. INTRODUCTION

1.1 Objectives.

In order to determine the suitability of the discrete event simulation language GPSS for modeling computer network structures likely to be encountered in command, control, and communication (C^3) systems, several example computer networks were simulated using this language. This report presents a survey of GPSS capabilities and peculiarities. Problems encountered in translating GPSS programs from one available version of GPSS to another as well as explanation of differences in simulation results are discussed. Results comparing performance of four ring network structures simulated in this study are also presented.

1.2 Background.

This is the first in a series of reports on the progress and results of AMSAA's work in creating new and using existing models in analyzing and predicting the performance of computer networks and their supporting communications.

1.2.1 Relation to C^3 Modeling. The study, construction, and validation of C^3 simulation models aids the work of the C^3 group by providing:

- data to support conclusions about proposed system concepts,
- tools for evaluating alternative configurations posed by requirements definition studies such as TOS CASE and ASAS FSD,
- the means to examine simultaneously computer system performance, network configurations, and imperfect communications,
- quantitative estimates of the effects of interoperability requirements upon the performance of the primary mission of a system and upon the supporting communications,
- data to augment that obtained from system testing, and
- estimates of the most difficult combinations of system inputs to satisfy, which can be used to guide test planning toward efficient and effective discovery of system deficiencies.

1.2.2 Motivation. The work reported here was motivated originally by an effort relating to TOS CASE in which varying approaches were proposed by contractors for modeling and simulating the combined computer processing and communication network for this system. Initially, a model of computer communications developed for the Air Force called

SACDIN was proposed for use but subsequently was rejected because it could not easily be modified to handle message routing in other than a tree connected hierarchical fashion. The contractor then proposed using a general purpose discrete event simulation language called GPSS to write a simulation model using the dialect of GPSS implemented on a Control Data Corporation 6600 computer.

To prepare AMSAA personnel for analysis of the validity of the anticipated TOS CASE simulation model, a study of GPSS was begun. Because only the UNIVAC dialect called GPSS 1100 was available to AMSAA personnel at the inception of this study, the question of syntactically and semantically correct translation of simulation programs (models) from one dialect of GPSS to another was raised. Much of the work reported here deals with answering this question.

1.2.3 Approach. In order to develop expertise in GPSS modeling of computer communication networks and to develop confidence in comparison of models written in one dialect of GPSS with those written in a different dialect, the team decided to translate known computer communication network models from one set of syntax and semantics into the other.

Models of computer communications networks written in GPSS were obtained from the open literature. Only those having a relatively simple structure coupled with published simulation results for comparison purposes were considered suitable for use in the study of translation from one dialect to another. Three of the models were written in an enhanced version of GPSS/360 for an IBM 360 series computer. The fourth model had been written in GPSS 1100 for a UNIVAC 1108 computer.

Initially, the study team was restricted to using only a UNIVAC 1108 computer; so three of the programs were translated from GPSS/360 into GPSS 1100, and several differences in output results were noted. Because the syntax of GPSS 1100 differs from that of GPSS/360 and its later dialect called GPSS/V, the correctness of the syntactic translation was studied. Careful desk checking of the translation by at least three independent programmers revealed no discernable errors, leading to a check of possible semantic differences.

Semantic differences are those due to the way in which the simulation command interpreter is actually executed. If the interpreter is written in a high level language (e.g., FORTRAN), the differences may be due to the manner in which the various subprograms are compiled; if the interpreter is written in assembly language, the differences may be due to different hardware characteristics such as word length.

Because the simulators both rely on pseudo random number generators to generate the stream of random events according to assumed probability distributions, it was first necessary to account for possible differences generated here. Because of differences in word length, the largest representable integers in the two systems are different. Hence, the two pseudo random number generators are inherently different. Initially, it was postulated that either one or both sets of pseudo random number generators may be exhibiting nonrandom behavior. To check this hypothesis some tests of randomness were performed on the generators, and these tests are documented in Appendix A. Even if the generators are sufficiently random, semantic differences in the way in which the generated numbers are subsequently used may be the cause of differences in the output results. Deterministic and identical tables of numbers supplied to both simulation dialects to guide event generation and flow in the models, coupled with detailed traces of activity in the models were then considered appropriate for finding differences. This approach required the availability of an IBM 360 computer system or equivalent. Ultimately, access to an IBM 360 computer system with GPSS/360 was obtained. The pseudo random number generators in both the GPSS/360 and GPSS 1100 simulations were disabled, and identical tables of pseudo random numbers (generated on a CDC 7600) were appropriately formatted and inserted into the two different dialect simulation models. As a result, certain semantic differences have been identified and are discussed in this report.

1.3 Organization.

Chapter 2 of this report discusses briefly the concepts of discrete event simulation and presents a short introduction to the GPSS language and some of its capabilities that are relevant to computer communications network simulations.

Chapter 3 introduces the ring network examples simulated in this study and presents results of those simulations. Lessons learned are also discussed.

The appendices include a summary of pseudo random number generator tests and their results, listings of GPSS programs for the computer networks used in this study, and a glossary of acronyms and abbreviations.

1.4. Summary of Conclusions.

Several conclusions were reached. They are:

(1) It is possible to correctly translate simulation programs from one dialect of GPSS into another, even though GPSS/360 and GPSS 1100 differ in both syntax and semantics. The GPSS/360 syntax uses fixed fields, and GPSS 1100 has a column free, easier to use format. Semantic differences are due to inherently different pseudo random number generators, the documented use of differing default conditions, and undocumented differences in function interpolation. Because of these differences, care should be taken when comparing output data from one dialect with that from another.

(2) GPSS/360 on the APG IBM 360/65 executes considerably faster than does GPSS 1100 on the ARRADCOM UNIVAC 1108, about four to seven times faster for the examples considered here and for other test cases that have been run.

(3) Both GPSS/360 and GPSS 1100 have attractive features for the discrete event modeling of computer/communications networks. Messages are easily modeled as dynamic entities called transactions. Language features are provided for causing message arrivals and other randomly occuring events. Equipment entities, such as transmitters, receivers, and message queues are easily modeled. Automatic collection and display of statistics on system performance are provided.

(4) Preliminary analysis of end to end message transmission delay data from simulations of four ring network link level protocols indicates that at low system loading there is no significant (order of magnitude) difference among them. The systems saturate or exhibit exponentially unbounded end to end delay times when sufficiently heavy loads are applied, and they do so in an order of increasing load consistent with previously published data.

2. SIMULATION WITH GPSS

2.1 Discrete Event Simulation.

System simulation using models having state variables that change state only at discrete instants of time, with time progressing in discrete increments, is called discrete event simulation. For a given interval of simulation time, points of event occurrence in discrete event simulation are both finite and countable, whereas in continuous system simulation the time of event occurrence is continuous. Because GPSS is a discrete event simulation language, any system being modeled in GPSS must be representable as a discrete event system. Doing so requires what may appear to be some degree of oversimplification, but simplification is acceptable if the model accurately reflects system behavior without necessarily reproducing exactly and completely the actual system operation. Since there are many different simulation languages available to the user the features that distinguish GPSS will be examined.

2.2 Features of GPSS.

One of the major advantages in using a language such as GPSS to simulate systems is the convenience afforded by the language [1]. Instrumenting a simulation model to collect data and compute statistics revealing the performance of system components of interest is a major task in constructing a system model. A large part of the statistics gathering is intrinsic to GPSS; hence the programmer need not ordinarily be burdened with this time consuming task. Along with its automatic data collection, GPSS allows the modeling of many of the significant characteristics of "real world" systems with much ease. The characteristics that are easily represented include dynamic entities, equipment entities, operational entities, data entities, and randomness considerations. Also subliminally used are the simulation clock and the event scheduling algorithm. A brief description of each of these factors follows.

2.2.1 Dynamic Entities. The dynamic entities, called transactions in GPSS, are used to represent a flow of some sort through the system. The transactions which "flow" through the model may either cause an activity or be the recipient of an activity. In other words, the transaction may itself cause the state of the model to change, or it may have any of its associated parameters changed. The altering of a parameter value of a transaction may in turn be used at another place (or time) in the model to effect changes to the state of the model.

2.2.2 Equipment Entities. Equipment entities are used in modeling components that have a specific action associated with them. Equipment entities include storages, facilities, and logic switches. Storages are used to represent entities that may have their activity dictated by one or more transactions, whereas facilities are used to represent entities that may have their activity dictated by only one transaction at a time. A logic switch is used as a binary state indicator, such as locked or unlocked, available or unavailable, and open or closed.

2.2.3 Operational Entities. The operational entities are used to perform a variety of functions. They provide for representation of system relationships, model activity control, and the basic structure of the model to name only a few. In GPSS the operational entities are blocks, queues, user chains, groups, and save values. Blocks are the basic unit of the model structure. Queues are generally used to monitor delays encountered by transactions at specific points in the model. User chains are used to alter the normal "flows" of transactions in a user defined manner. Transaction "flow" may be controlled on the basis of group membership, where group membership indicates a certain relationship existing between transactions in the group. Savevalues are used to store information at certain locations in the model.

2.2.4 Data Entities. Data entities are used to input data to and output data from the model as well as to represent certain data relationships. The data entities available to the GPSS user include functions, variables, and tables. Functions are a means of entering distributions of various types to the model. The distributions may represent real system data or they may merely specify standard distribution forms that may be necessary to the simulation. Variables are used to represent system data relationships. Tables are included as a means of extracting data from the model.

2.2.5 Pseudo Random Number Generators. In addition to the various entities that can be modeled, the GPSS programmer has a number of pseudo random number generators available to him to aid in the simulation of randomly occurring events. The pseudo random number generators are actually deterministic, of course, but this offers one distinct advantage--reproducibility of simulation runs for program debugging purposes using the same sequence of numbers from run to run.

2.2.6 Simulation Clock and Event Scheduling Algorithm. The simulation clock and the event scheduling algorithm are related concepts.

The GPSS simulation clock does not advance time in fixed unit increments. Instead the simulation clock is advanced only when the next event is scheduled, and is advanced to that next scheduled time directly. Event scheduling is effected by scanning one of several "event chains," or ordered lists of transactions. After the appropriate chain is scanned, processing of transactions occurs on the basis of scheduled departure time, currently assigned priority, and time resident on the chain. After all events that can take place at the current simulation clock time have occurred, the simulation clock is advanced to the next scheduled event occurrence determined by a scan of the future events chain. Simulation continues in this fashion until an event occurs that terminates the simulation at some desired point.

2.3 Comparison of GPSS/360 and GPSS 1100.

The two dialects of GPSS available to the study team were IBM's GPSS/360 [2] and UNIVAC's GPSS 1100 [3]. The IBM version of GPSS executes on the APG IBM 360/65 computer system, and the UNIVAC version executes on the ARRADCOM UNIVAC 1108 computer system. These two versions of GPSS are distinct implementations of the same discrete event simulation concept, but there are a number of differences between them as discussed below.

2.3.1 Syntax. Both versions of GPSS have the same basic block structure, but syntax varies considerably between the two. UNIVAC GPSS 1100 uses a relatively free form input format in its statement specification language. Similar to the UNIVAC Assembler input statement formats, various fields appearing in the line image of a GPSS 1100 source statement are not *column dependent*, are simply separated by one or more blank spaces, and in some cases are not required to appear in a specified order. In IBM GPSS/360 the fields of a source statement must appear in specific column locations in the line image. For example, the location field used to identify a specific statement for later symbolic reference must begin in column two and not extend past column six. This places a five character limitation on statement names or identifiers. Identifiers in GPSS 1100 can be more than five characters in length, resulting in the ability to use more descriptive location names.

2.3.2 Function Definition. Another difference between the languages is in the area of user defined versus simulator supplied functions. Both simulators provide several callable pseudo random number generators with which simulator supplied uniform distribution functions are generated and for which the user need only supply the end points. In order to specify an exponential probability distribution function or a Gaussian distribution function in IBM GPSS/360, the user must supply a finite set of x and y coordinates that, coupled with simulator supplied linear interpolation, approximate the desired distribution function. Depending on desired accuracy, approximations of 24 to 60 or more points are typically used. The UNIVAC GPSS 1100 simulator supplies uniform, exponential, and Gaussian distribution functions as built-in components of the language. To invoke the exponential or Gaussian distribution, the GPSS 1100 user need only reference them with appropriate parameters (a mean value for the exponential function and a mean and variance for the Gaussian function). As with the IBM GPSS implementation, the user can define any other desired functions by specifying appropriate sets of points.

2.3.3 Memory Allocation. Because IBM GPSS/360 allows the programmer to specify either halfword or fullword values for parameters and savevalues, the programmer can save some memory space for allocation to other purposes. This represents an advantage over the GPSS 1100 version. The assignment of halfword parameters and savevalues normally might be used only minimally by most modelers. A second and fairly small benefit is that smaller models run faster. Perhaps there are only a few instances where a decreased run time may be noticeable, but in these few instances it may be a large advantage. The feature of variable word size for parameters and savevalues gives GPSS/360 greater flexibility than GPSS 1100.

2.3.4 Function Interpretation (Interpolation). The two versions of GPSS differ slightly in the way that they perform interpolation in user defined functions. For example,a continuous function may be defined with x-coordinate values of 0 and 1000 and corresponding y-coordinate values of 1 and 6, respectively. This defines a straight line segment between the points (0,1) and (1000,6). Now, given that the x value is to be determined by some random number generator with values ranging from 0 to 999, and that both interpreters operate by truncation rather than rounding, the functions can then yield results of 1,2,3,4, or 5 with equal likelihood. Since the representation of single-precision floating point numbers in IBM 360 computers uses a 32-bit hexadecimally normalized format and in UNIVAC 1100 computers a 36-bit binary normalized format, the representation of certain fractions is not exact.

The expression of certain numbers was found to be a problem in the above example. It was found that for an x value of 200, the IBM simulator returned a y value of 2--the result that one would expect. The UNIVAC simulator, however, returns a value of 1 for the same input x value. Further investigation found that both the IBM and UNIVAC versions returned the correct y value of 2 for an x value of 201, and the correct y value of 1 for an x value of 199. The problem again arose in the evaluation of x coordinates of 400, 600, and 800.

One reason for the discrepancy may be attributed to the order in which arithmetic operations are carried out in the interpolation process. Since truncation is used, the order of operations is important. For example, letting (x_1,y_1) and (x_2,y_2) be the endpoints of a continuous straightline function in which intermediate interpolated values are desired, the interpolated value y is given by:

$$y = [(y_2-y_1)/(x_2-x_1)]\cdot x + [(x_2y_1 - x_1y_2)/(x_2-x_1)]$$

$$= mx + b, \qquad \text{where}$$

$m = [(y_2-y_1) / (x_2-x_1)]$, and

$b = y_1$ if $x_1 = 0$.

In the case considered here, $b = y_1$.

Two of the possible combinations for ordering operations in the computation of y are:

Approach 1:

```
Step 1:  set $m := [(y_2-y_1)/(x_2-x_1)]$
Step 2:  set $z := m \cdot x$
Step 3:  set $y := z + b$
Step 4:  set $y :=$ integer $[y]$ , i.e. truncate fraction.
```

Approach 2:

```
Step 1:  Set $z := (y_2-y_1) \cdot x$
Step 2:  Set $w := z/(x_2-x_1)$
Step 3:  Set $y := w + b$
Step 4:  Set $y :=$ integer $[y]$ .
```

In certain instances such as $(x_1,y_1) = (0,1)$ and $(x_2,y_2) = (1000,6)$ and $x = 200$, Step 3 of Approach 1 produces 1.9999999926_{10} for the UNIVAC single precision floating point format and 1.9999990463_{10} for the IBM single precision floating point format. If the order of operations in both IBM and UNIVAC GPSS implementations corresponds to Approach 1 (and at least IBM GPSS/360 documentation [22] pp. 75 & 205 seems to so indicate), then the y value returned in both systems (after truncation) would be unity. Using Approach 2 with the same data items as above, the result is the integer value y=2 for both the UNIVAC and and IBM interpolation schemes. Empirical results using the above data items in both GPSS implementations produces interpolated integer values of y = 1 for the UNIVAC implementation and y = 2 for the IBM implementation, indicating that perhaps the available IBM GPSS documentation does not accurately reflect the actual ordering of operations, or that the documentation available to the study team does not include all possible change notices. The UNIVAC implementation would appear from this single sample to accurately follow the operation ordering stated in the IBM documentation. In any event a likely cause of observed differences in GPSS function interpolation between the two implementations is due to different (nonequivalent) orderings of finite precision floating point arithmetic operations.

Determining the exact cause of the differences would require laborious and time consuming detailed examination of the assembly level machine code for the two GPSS implementations, which is beyond the scope of this study. The most important fact has been ascertained: namely, exact and correct syntactic translations of GPSS programs between IBM GPSS/360 and UNIVAC GPSS 1100 can produce differing output values that are caused by semantic differences in the implementations of interpolation.

2.3.5 Miscellaneous Differences. Miscellaneous differences between GPSS/360 and GPSS 1100 include the simulation clock starting time and the calculation of standard deviations in the standard statistical output. The IBM version of GPSS starts its simulation clock at time one, while the UNIVAC version starts its simulation clock at time zero. This is a minor difference, but one whose effect can be seen when a model's transaction routing is a function of absolute simulation clock time. The UNIVAC clock can be aligned with the IBM clock by specifying that no transaction enter the model before time one. Differences in calculated standard deviations, though small, were observed when start time, and the generation and movement of all transactions were forced to be identical in deterministic models. The reason for these standard deviation differences is not apparently due to one version producing best estimates of standard deviation and the other not doing so, and the exact reasons for these modest differences are not yet understood.

2.3.6 Random Processes. One point to be considered when running stochastic simulations is whether processes to be modeled as random can be modeled acceptably. Each of the two versions of GPSS offers pseudo random number generators to aid the modeling of stochastic processes. IBM GPSS/360 offers one such generator replicated eight times. Hence, a user can implement up to eight distinct sequences of random numbers. The sequences will be identical initially unless the user inputs a seed different from the default value to one or more of the generators. UNIVAC GPSS 1100 offers ten distinct pseudo random number generators. The generators are of the same type (either linear or mixed linear congruential) but use different seeds and multipliers. Statistical properties of pseudo random number generators for both GPSS versions were studied to determine whether the generators are random enough, and details of that study are presented in Appendix A. In summary the pseudo random number generators are generally random enough for use in the ring network simulations discussed in the next chapter.

2.3.7 Run Time. One last consideration of the differences between IBM GPSS/360 and UNIVAC GPSS 1100 is simulation execution time (or run time) and its corresponding cost. The CPU time for four ring network models using the IBM GPSS simulator was from one fourth to one tenth of that required to execute the same models using the UNIVAC GPSS simulator. For example, GPSS simulation of the DLCN model described in Chapter 3 required 4 min. 16 sec. of CPU time for the IBM version and 30 min. 33 sec. of CPU time for the UNIVAC version of the model using identical system parameters. For this example the UNIVAC version runs about seven times slower than the IBM version. There is apparently a significant speed (and hence, cost) advantage in running GPSS/360 models over the GPSS 1100 models.

Turnaround time, measured using wall clock time, was also generally better on the APG IBM 360/65 than on the ARRADCOM UNIVAC 1108 when running corresponding GPSS simulations for the four ring networks considered in Chapter Three. Wall clock time includes a measure of system congestion, and to the programmer fast turnaround is usually of interest. Sample simulations run as the only batch job on the system at times when time sharing demand service was cut off indicate similar ratios of wall clock time. Sobel[7] was plagued by extraordinarily long run times under similar system loading conditions on a UNIVAC 1100/42 system. Simulation runs that finished normally on the APG IBM 360/65 in an hour of wall clock time terminated abnormally on the much faster UNIVAC 1100/42 system in approximately four hours of wall clock time on an essentially empty system, where abnormal termination was caused by the need to exceed the programmer specified run time limit. Although the UNIVAC 1108 is a faster computer than the IBM 360/65 according to Schriber [1] the UNIVAC GPSS 1100 simulator appears to have a far less efficient implementation than does the IBM GPSS/360 simulator. Models executed from four to seven times faster in the IBM version. In addition, comparison of wall clock times for the four ring network simulations revealed that the IBM 360/65 system gives from two to three times better turnaround than does the UNIVAC 1108 system. This may not be true in all modeling situations, but for the rather simple ring network structures studied IBM GPSS/360 is more efficient than UNIVAC GPSS 1100. This conclusion is, of course, system configuration and site dependent.

2.4 Suitability.

2.4.1 Ease of Model Implementation.

The first factor in determining the suitability of GPSS for modeling computer communications networks is the ease of model implementation. Each block in the structure of a GPSS model may represent a separate action block in a flowchart of the system being modeled. For instance, the process of capturing a facility for some length of time and then relinquishing control of the facility requires three GPSS blocks: one to seize the facility, one to advance the clock, and one to return the facility to its previous state. This is considerably simpler to specify in GPSS than it might be in many other programming languages in which it may be necessary to write one routine to implement each of the three GPSS blocks. The event processing routines are intrinsic to the GPSS language, so the user need not be bothered by the possibly unpleasant task of describing each action in detail.

2.4.2 Understandability.

Another factor in the ease of model implementation in GPSS is this language's choice of block names which aids understandability. The process of obtaining control of some facility is written as SEIZE "facility" in GPSS. The SEIZE block is then a model statement that can be readily understood by managers as well as programmers. The majority of blocks in GPSS are named in such a way that the block name describes the block function.

2.4.3 Standard Statistical Output. Another advantage in building models with GPSS is the standard statistics gathering intrinsic to and aided by the language. Statistics such as queue times, storage contents, and facility utilizations are all collected automatically by the GPSS simulator. These items, along with a large number of other useful statistics, are printed in a standard statistical package in the output report of the program. Additional information concerning the model run can be obtained by the inclusion of user defined tables in the output report.

2.4.4 Optional Output. As optional output, TRACE and PRINT blocks are available to aid in the debugging of a GPSS program. After all known bugs have been removed from the simulation model, the programmer may specify optional output formats and histograms as well to make the output understandable to the nonspecialist.

2.4.5 Level of Detail. An additional consideration in assessing the appropriateness of GPSS for computer communication network models is the level of detail permitted in the models. If the modeling objective is to develop an exact detailed replica of the real world system, then it is doubtful that GPSS would be a suitable language. If, however, the objective of the model is to gain general insights into how a system will perform under various circumstances, then GPSS could be a suitable language. Because there are memory space limitations on the size of the GPSS program, some simplifications must be made as a trade-off. In deciding whether to model in GPSS, the analyst must determine whether the amount of simplification required is acceptable. Language features permit the programmer to command reallocation of the available data storage space among the competing entities invoked by block specifications. However, large models (i.e., those with large numbers of blocks or large numbers of simultaneously active transactions) can easily exceed the available storage on the machine executing the GPSS simulator. In such cases the programmer may be forced to reduce the level of detail simulated in order to get his model to run at all in the existing hardware/software environment.

Similar decisions and limitations are faced by analysts and programmers in every language chosen for performing simulation. In some languages the ability to call operating system service routines or other library routines may be more easily performed than in GPSS. Resolving problems at acceptable cost in time and effort is the key issue and must be traded against ease of simulation model implementation directly available from language features and level of detail required.

3. COMPUTER COMMUNICATION NETWORK MODELS

3.1 Network Concepts and Terminology.

Computer communication networks are essential components of military C^3 systems. Computer communication networks permit users to access resources such as hardware units, software packages and data files in a remote computer system. One can view the structure of a computer communication network as being partitioned into two parts, a communication network (sometimes called the communication subnetwork) and a user resource network[4].

3.1.1 Communication Network. The communication network comprises the switching computers (or nodes) and the communication channels. Its function is to deliver messages from one node to another.

3.1.2 User Resource Network. The collection of terminals and computing resources comprises the user resource network. These resources are connected to switching nodes and communicate with each other by way of the communication network.

3.1.3 Hosts, Protocols, and Network Function. The computer systems in the user resource network are called hosts, and a set of protocols is implemented in the operating system of each host. These protocols are procedures to initiate, maintain and terminate software communications via the nodes of the communication network. A host computer may accept jobs (such as requests for processing, data base queries or updates, etc.) from local or remote users. Remote jobs are received as messages from the communication network, and require extra processing time for protocol handling. When processing of the remote task is complete, the results are repackaged as a message (or a set of related messages) and are returned to the remote users via the communication network.

3.1.4 Message Switching. The basic technique by which messages are delivered from source node to destination node in a communication network is called message switching. In this technique a message entering the network is first passed to its origin node where it may be stored while it waits for route selection according to some routing algorithm and where it may queue for its outbound communication channel. When the channel becomes free, the message is transmitted to the next node along its route to the destination, and the above process is repeated until the message is delivered to its destination node.

3.1.5 Packet Switching. A modification of message switching is a technique called packet switching wherein each message is decomposed into maximum length disjoint subsets called packets. Each packet is identified for later message reassembly, and each can be routed independently through the communication network.

3.1.6 Performance Measures. The total elapsed time from the arrival of a message at its source node to the successful delivery of this message to its destination is called end-to-end delay and is an important performance measure of both message and packet switched networks. Factors influencing this performance measure include assumed (or actual) message arrival and message length statistics, routing algorithms, channel service and error rates, resource contention and assigned priority classes, and queueing and buffering delays enroute. In order to minimize end-to-end delay, designers need tools with which to predict its mean, variance, and distribution subject to sets of input parameters. Other performance measures and the effects of design parameters must also be analyzed in order to determine quantities such as optimal finite buffer size, channel utilization, and system throughput (i.e., messages/unit time).

3.2 Network Modeling.

Queueing network models have been used extensively in the performance analysis of message switched (or packet-switched) communication networks.

3.2.1 Analytic Models.

Closed form analytic models, when available, are advantageous in that they can lead to low computational cost predictions. Exact analytic analyses are restricted to certain classes of simplified models [5], and results for general models with more complex features, such as adaptive routing algorithms and finite buffer space, are not yet available.

3.2.2 Simulation Models.

Discrete event simulation models have been used both to verify the adequacy of simplified analytic models and to provide performance analyses in cases as yet too complex for adequate analytic models. The generality of simulation models is paid for in higher computational costs and generally greater computer execution times than may be required for evaluating analytic models. If partial analytic results are available, mixed analytic and simulation models help to reduce simulation costs. In many cases the system description parameters such as non-Poisson arrival statistics and state transition probabilities are either not available or not directly useable in the analytic models; whereas enough information may be available to implement a discrete event simulation whose input is a list of measured arrival events from some actual systems.

3.3 Network Topologies.

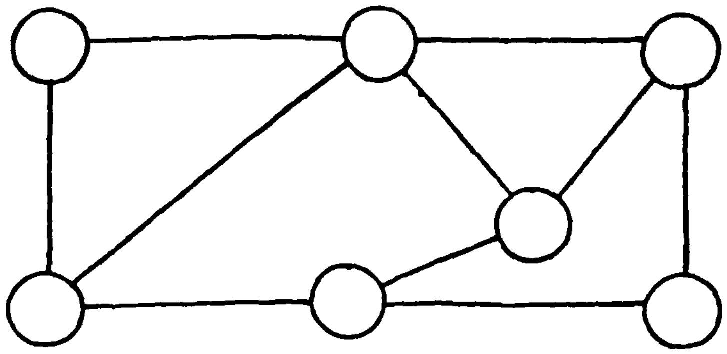
Figure 3.1 shows three basic topologies commonly found in computer communication networks: the mesh, the tree, and the ring; variations of these also commonly occur. Internetwork configurations wherein nodes in one topological network structure act as gateways to other (or even the same) topological network structures are also frequently encountered.

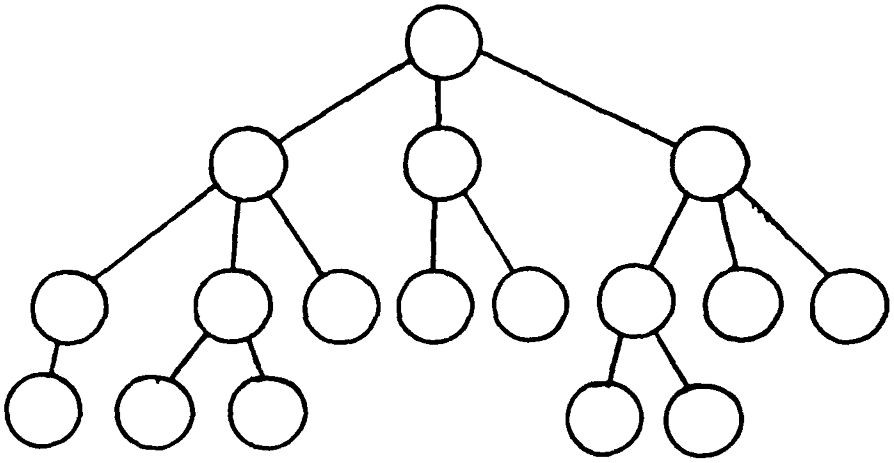
3.3.1 Mesh.

A mesh connection of nodes is characterized by a connectivity generally greater than or equal to two at each node so that at least a subset of nodes can select alternate routing paths between source-destination pairs.

3.3.2 Tree.

A tree connection is characterized by a hierarchical structure in which the message path between two nodes at the same level in the tree must pass through a common ancestor node at a higher level in the tree.


a. MESH


b. TREE

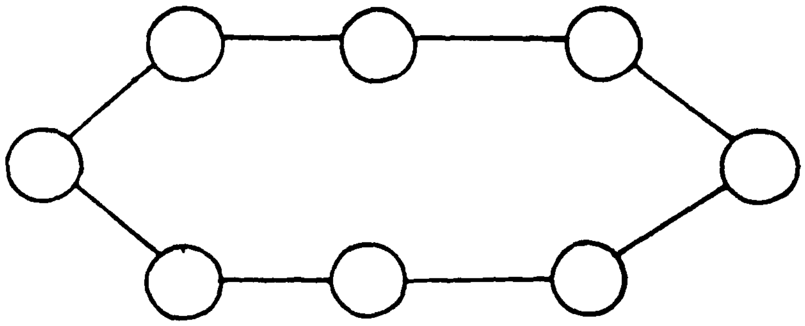

c. RING

Figure 3.1: Some Computer Communication Network Topologies

3.3.3 Ring. The ring is characterized by a node connectivity of exactly two and a unidirectional transfer of information around the communication links. A message going from a given node to its predecessor node in the ordering implied by the direction of information transfer on the ring must pass through all the nodes on the ring to reach its destination.

3.3.4 Variations. Topological variations in mesh connections range from minimal to maximal connectivity, and to structures resembling tree structures with cross connections between a subset of nodes in different branches but at the same level in the tree. The principal topological variation for ring (or loop) networks comprises two or more rings (usually passing messages in opposite directions) for greater reliability and increased throughput.

3.4 Ring Network Structures Considered.

Because the routing structure of rings and loops is deterministic and simple, and because GPSS models of both message switched and packet switched ring networks are readily available in the literature [6,7], this network topology was chosen for further investigation in the simulation study of computer communication networks presented here. Validation of the simulation models and comparisons with prior work of others are possible for this topology because earlier simulation results are available in the open literature [8,9], and this provides greater documentation and insights than are usually available for more complex topological structures.

A loop network is sometimes distinguished from a ring network according to whether the communication access control protocol is centralized (loop) or distributed (ring). Some authors refer to loops and rings interchangeably, including those who have designed loop networks with distributed control mechanisms [8,9,10,11,12].

Four basic types of single loop networks have been proposed in the literature, namely, the Newhall, Pierce, DLCN, and Playthrough structures. These loop networks are distinguished by their transmission control and link access mechanisms.

3.4.1 Newhall. The earliest loop structure was proposed by Farmer and Newhall [13], and is commonly referred to as a Newhall loop. In this structure a single control token is passed from one loop interface to the next until it reaches a node with a message to transmit. That node temporarily seizes the control token and starts transmitting its (variable length) message to an addressed destination node. Intervening nodes pass this message to the destination node which, according to varying implementations, either copies the message into its arrivals buffer or removes the message from the loop. For error checking purposes the message sometimes is permitted to circulate to the receiver portion of the source node, which then performs a consistency check and removes the message from the loop. Also, depending upon implementation, the source node currently in possession of the control token may transmit one or more variable length messages before relinquishing control of the loop by passing the control token to the next node in sequence. Only one source node may transmit at a given time, and all other potential source nodes must wait to transmit queued messages until they receive the control token. Several experimental and commercial loop communications systems for interconnecting computers and components have been based on minor variations of this link level protocol structure (e.g., [14, 15]).

3.4.2 Pierce. The Pierce loop [16,17,18,19] divides communication space on the loop into an integral number of fixed size slots, called packet frames, into which data packets can be placed. To send a message, a node segments the message into fixed length packets, appends necessary overhead information to identify both the packet's number and the message to which it belongs, places each packet into the next available empty slot passing the node, and marks the slot as full. As this full message packet proceeds toward its destination, the other nodes along the route examine the header information in each packet frame to ascertain which of them is the addressed destination. The destination node, having recognized its address, copies the data being received and either fills the slot with new outbound information or passes this now empty slot to the next node. Incorporated into the loop is a single special control node that maintains time slot synchronization for the loop and prevents buildup of undeliverable packets. The header of each packet passing through this control node is marked; if a packet tries to pass through this control node a second time, it is typically destroyed, creating an empty slot.

3.4.3 DLCN. The link level transmission scheme for the distributed loop computer network (DLCN) [6,8,9] uses a shift register insertion technique to place variable (but hardware restricted) length messages onto a ring. Two shift register buffers are used; one is a variable length delay buffer that receives data from the predecessor node, and the other is a fixed length shift register that contains data to be placed onto the ring at the present node interface.

A message arriving for transmission at a given node waits in the output buffer until end of message is detected for the data message passing through that node from predecessor to successor nodes. When this event occurs, new incoming data from the predecessor node is routed into the delay buffer, and data in the output buffer is shifted out onto the ring, thereby splicing the waiting message at this node between two messages already in transit on the ring. In other words, so long as there is enough space available in the delay buffer to hold an incoming message, precedence is usually given to transmitting a newly arrived or already waiting message at the present node ahead of an incoming message already on the ring. This technique tends to minimize waiting times for messages to be placed onto the ring at the expense of randomly delaying transmitted messages en route to their destinations. The maximum length message, which is in effect a variable but maximum length packet, is fixed by the length of the delay buffer at each node. When a message reaches its destination it is removed from the ring by that node. If the message is received correctly, a high priority acknowledgement message is placed on the ring by the destination node, addressed to the source of the received message.

Presumably, a message whose source or destination fields are corrupted will be error checked in such a fashion as to prevent the wrong destination from acknowledging correct receipt of the message. As with receipt of a negative acknowledgement, lack of a positive acknowledgement after some appropriate time period (called a time out) could cause the source node to retransmit the data message. A message unclaimed by its destination would also presumably be removed from the ring when the source address is recognized by some source node as part of its check and forward operations. Since DLCN uses a distributed control mechanism, there is no central controller to perform any of these functions.

3.4.4 Playthrough. The Playthrough mechanism for distributed control of ring networks [10,11,12] is a check and forward link level control protocol that provides for simultaneous transmission of multiple variable length messages of any length. Control is completely distributed, and data and control messages both share the ring. Control is based on a special synchronizing message (or token) called GO that differs from the Newhall synchronizing token in two ways: first, GO precedes rather than follows data messages, so that it can continue around the ring seeking new messages to activate; and second, GO circulates perpetually despite the presence of other traffic. This perpetual circulation is achieved by giving GO a higher priority and allowing it to preempt temporarily any data message it overtakes. Thus GO appears at times to travel inside data messages, or in golfing terms, to "playthrough." The protocol bears the name of this distinctive feature.

When GO arrives at any node with a message to send, transmission may begin if there is a free path to the destination. To implement this rule without collisions, other control messages precede and follow the data message to update the other nodes about changes in loop status. Thus the nodes must be able to recognize control messages and maintain a modest amount of local information about the ring. In order to propagate such status information, the update control messages play through any data messages they encounter. Although the update messages are synchronized by GO, their even higher priority causes them to precede GO so that each node has the correct status information before GO arrives.

Some operational aspects of this ring are worth noting. Data messages can be preempted only at their sources. This means that there is no store and forward phenomenon or buffering delay en route to the destination, except for a small fixed amount at each node. The delays from preemption are brief because the intervening control messages are short. Hence, the primary message delay is due to queueing at the source.

Except for GO which continues traveling, each control message makes exactly one complete circuit of the ring and is removed by its source. This permits acknowledgements from the destination node to ride for free on returning control messages and to avoid queueing delays. In addition, control messages complete the round trip in a fixed time that can be determined dynamically. This enables a very accurate timeout mechanism to be used for error detection and for capture and removal of unacknowledged or corrupted control messages.

3.5 GPSS Models of Ring Networks, Program Modifications, and Corrections.

Three network models written by C.C. Reames [6] in GPSS/360 were obtained through the assistance of Professor M.T. Liu of The Ohio State University. These programs for the Newhall, Pierce, and DLCN single ring computer networks were then modified to run under GPSS/360 on the APG IBM 360/65. Listings of these GPSS/360 programs can be found in the appendices of the PhD dissertation by Reames[6], pages 178 to 194. Short excerpts showing our modifications to these programs are included in Appendix B. They were also translated into GPSS 1100 for execution on the ARRADCOM UNIVAC 1108. GPSS 1100 listings can be found in Appendix C; the line for line comments are the same as those for the IBM versions in [6] and were thus omitted here.

A GPSS 1100 simulation program appearing in [7] for the Play-through protocol ring network, found here in Appendix C, was modified and corrected slightly and also translated into the GPSS/360 version found in Appendix B. In this case, line for line comments are included in the GPSS 1100 and the GPSS/360 versions to align the translations.

Several modifications to the original GPSS/360 and GPSS 1100 programs were made; some changes were necessary to allow the models to execute under GPSS/360 and/or GPSS 1100, and some were made to align the assumptions concerning message routing and error handling and to correct minor errors.

3.5.1 Changes to the Pierce Model. The GPSS/360 (enhanced) Pierce network simulation program referred to the absolute clock standard numerical attribute, which is not available directly in either of the available versions of GPSS/360 or GPSS 1100. Hence, additional code to effectively simulate the absolute clock facility was placed into the Pierce network simulation programs between labels LASTP and PATW.

3.5.2 Changes to the DLCN Model. The original DLCN program [6] attempts to simulate the effects on system loading and total message transit time (or end to end delay) caused by noise corrupted messages that include one or more erroneous characters. If the message (i.e. transaction) is marked as being received in error, it is discarded by the destination node, a negative acknowledgement is sent to the source node, and the message is placed at the front of the source node message queue for retransmission. Unfortunately, the implementation of this feature incorrectly counts the erroneous message as a successful reception (in terms of the statistics for end to end delay, and queueing time), and then resets the corresponding message's time in system to zero so that it appears and is counted in the statistics as a newly arriving message that encounters hardly any queueing time thereby slightly skewing the output statistics. Because of this approach to handling the simulation of erroneous messages with a mean character error rate of one in ten thousand, mean total transmission time for all messages handled by the network when errors are permitted to occur is about 10 percent lower than the mean total transmission time found when no errors occur, as seen in Figure 3.2. Such a result is counterintuitive and slightly incorrect. Because the other ring network simulation programs have no provisions for handling messages with errors in them, the character error generation facility in the DLCN program was disabled, resulting in a version referred to as DLCNNE for "no errors". This allows a more uniform comparison of simulation results for the different ring network protocols and removes an apparent cause of skewed results in the total time statistics for DLCN.

3.5.3 Changes to the Playthrough Model. Because of the rather complex and specific ordering in which messages must be placed on the communication links, the Playthrough simulation program maintains its own user chains, which are in effect user controlled transaction queues. The user chain is scanned in first-in first-out order to locate the first message in the queue having a free path to its destination. If one is found, that message (transaction) is removed from the queue and the remaining entries are left in their original order in the queue. This is accomplished by circularly shifting the queue entries and examining the leading entry until either a message with a free path is found or until the queue is restored to its original condition given the number of elements on the queue. Sobel's original implementation for certain queue conditions miscounted the number of circular shifts by one so that reordering of the queue after removal of an interior entry left one element out of position, resulting in occasionally increased waiting times for some transactions. A minor modification to the logic governing chain reordering corrected this problem.

The Playthrough message destination assignment scheme was modified to match that found in the Reames models so that the distribution of destinations is uniform. Sobel's original scheme generated message destinations skewed toward shorter distances.

3.6 GPSS Ring Network Simulation Results.

This section describes the results of running both IBM GPSS/360 and UNIVAC GPSS 1100 programs for the various ring network models. It was assumed that published data [9] were based on the same startup and run termination conditions found in the Reames programs from [6]. The Pierce model uses a startup of 250 messages to preload the queues and initialize the system, and then accumulates statistics on the successful transmission of 1200 additional messages. The Newhall model uses a startup of 200 messages for initialization and then accumulates statistics over 1000 additional messages. The DLCN and Playthrough models both use a startup of 100 messages and accumulate statistics on 1000 successfully transmitted messages. Without detailed statistical analyses of these startup parameters to determine if steady state has actually been reached, these seemingly arbitrary but intuitively justifiable choices lead to acceptable

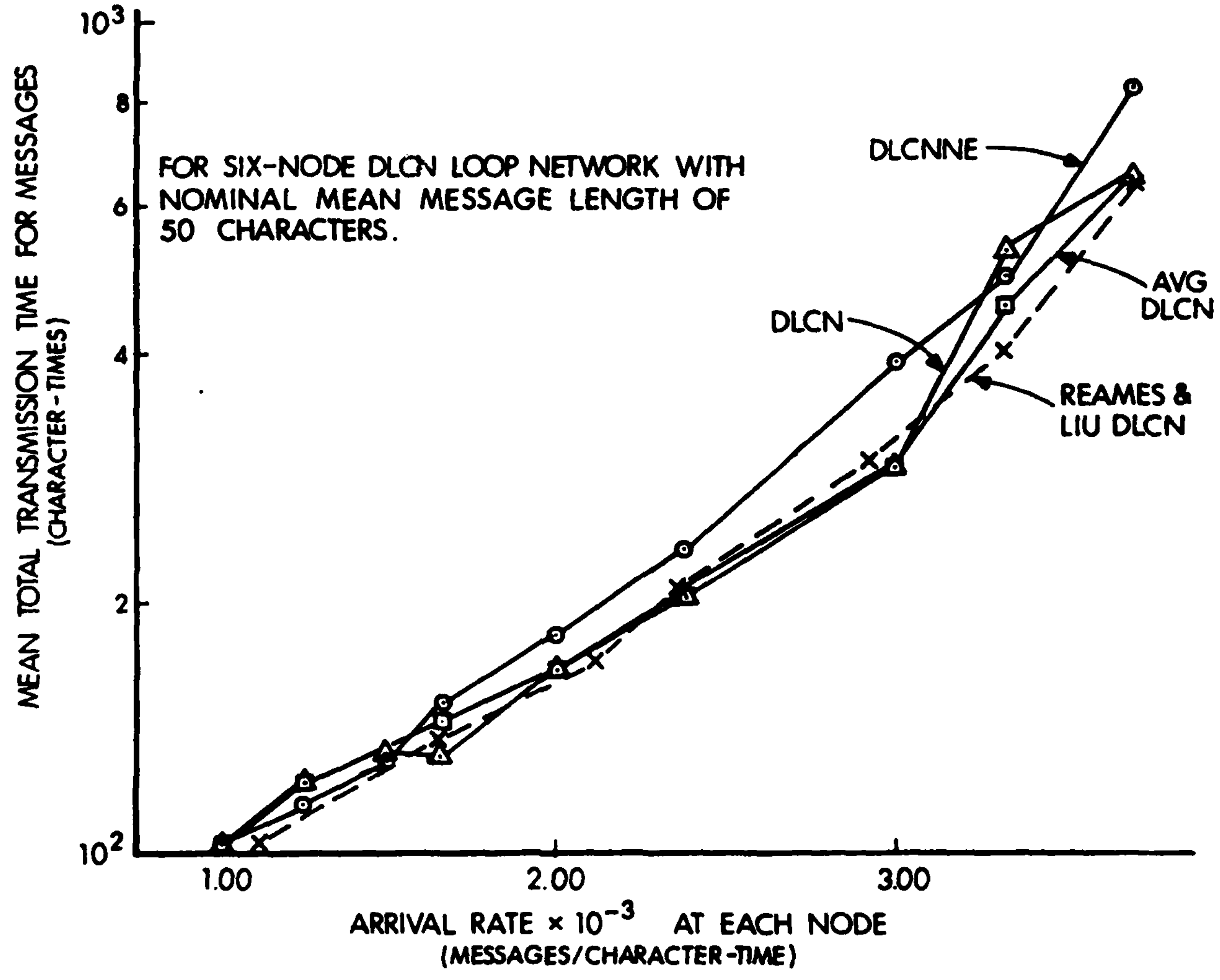


Figure 3.2. Comparison of DLCN Output Data with those for DLCNNE.

qualitative results only if one is interested in gaining an idea of relative performance differences. A check of startup conditions plotting relative changes in the mean of output parameters was made for DLCN and Playthrough indicating that 100 terminated messages seems to be sufficient for the warmup period. However, if one wishes to draw statistically valid inferences from the simulation results, one should use formal statistical tests while collecting the data.

Three important areas of concern are (1) starting criteria for data collection, (2) stopping criteria, and (3) determining to what degree the data are correlated. Starting criteria are concerned mainly with determining at what point the simulation closely approximates steady-state. Stopping criteria determine when (how soon) it is statistically safe to stop collecting data and still be able to draw conclusions with the required level of confidence. Correlated data yield less information about a system per observation than if all data were independent. To compensate for this lower average informational content, one must collect more data. Later simulations of DLCN by itself for example [21] take cognizance of these items. Although statistical validity of simulation results was not the main concern of this study, it must be a major consideration of any production oriented simulation study on whose results decisions are to be based.

3.6.1 Message Interarrival Time and Length Distributions. Tests of the correctness of generated exponential distributions in both IBM and UNIVAC simulations were performed. Because message arrivals at each network node (from its attached component) are assumed to be governed by a Poisson process with identical parameters at each node, plots of actual interarrival times were made to see if they resemble exponential distributions and to see if those generated by the UNIVAC intrinsic exponential function are similar to the IBM user defined exponential function. One such example plot showing count of the number of messages versus corresponding interarrival time, where interarrival times are grouped into ten unit intervals, is shown in Figure 3.3. The mean interarrival time is 300 character times at each node; for the six node system considered here the system's mean interarrival time is 300 divided by 6.

A sample plot of the count of the number of messages versus corresponding message length is shown in Figure 3.4. Each generated message has nine characters of overhead information added to its length, and each frequency count was accumulated over a ten character interval after overhead information was appended; hence, the first interval counts messages of length between nine and ten characters only thus skewing the plot from a true exponential. All of the ring network simulation programs considered here use an approximate exponential distribution for generating message lengths truncated at a maximum of 500 characters because of the DLCN hardware defined delay buffer limit of 512 characters including overhead.

Overall, the UNIVAC and IBM generators produce similar results for exponentially distributed interarrival times and message lengths.

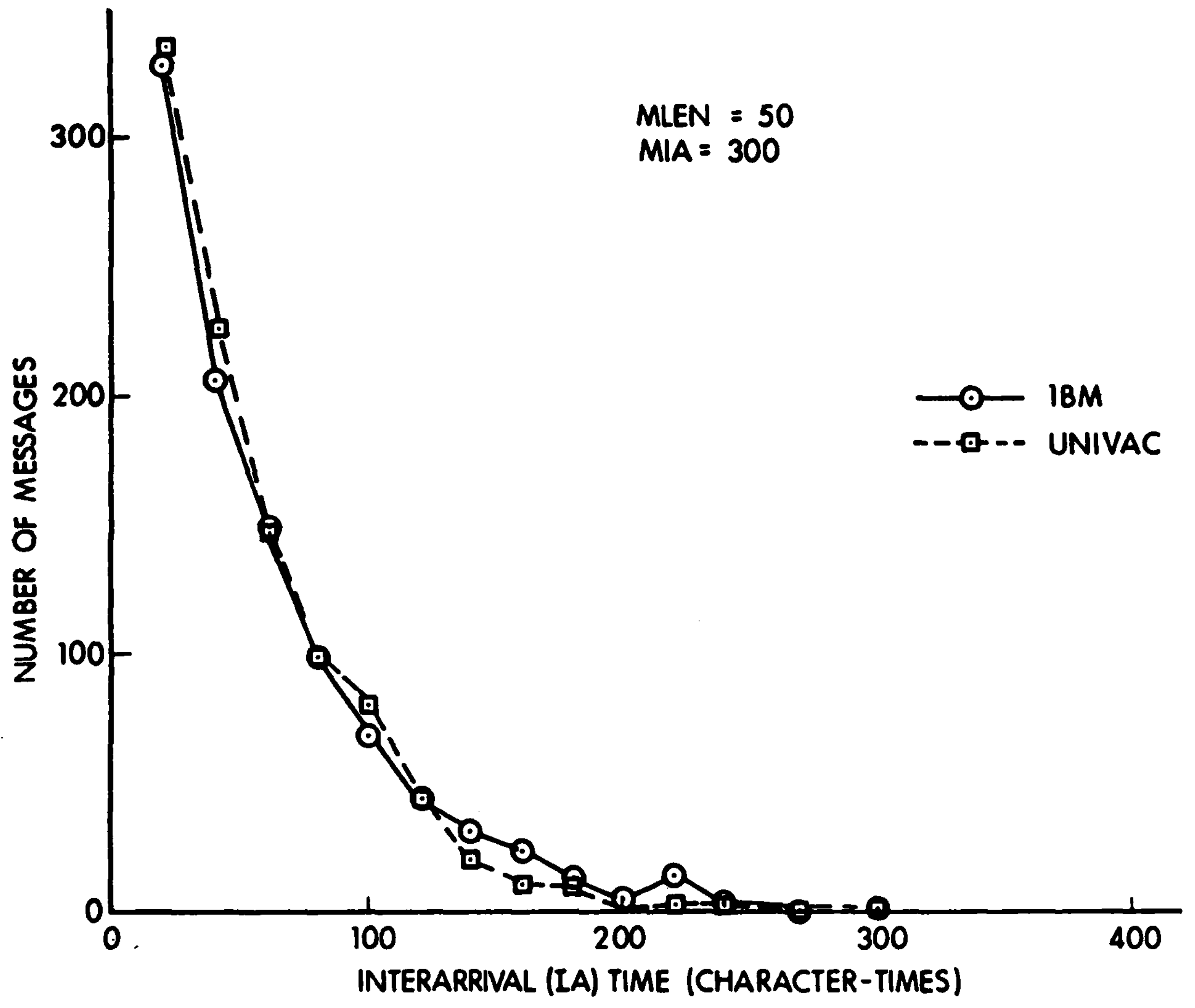


Figure 3.3. Generated Arrival Distribution (DLCN).

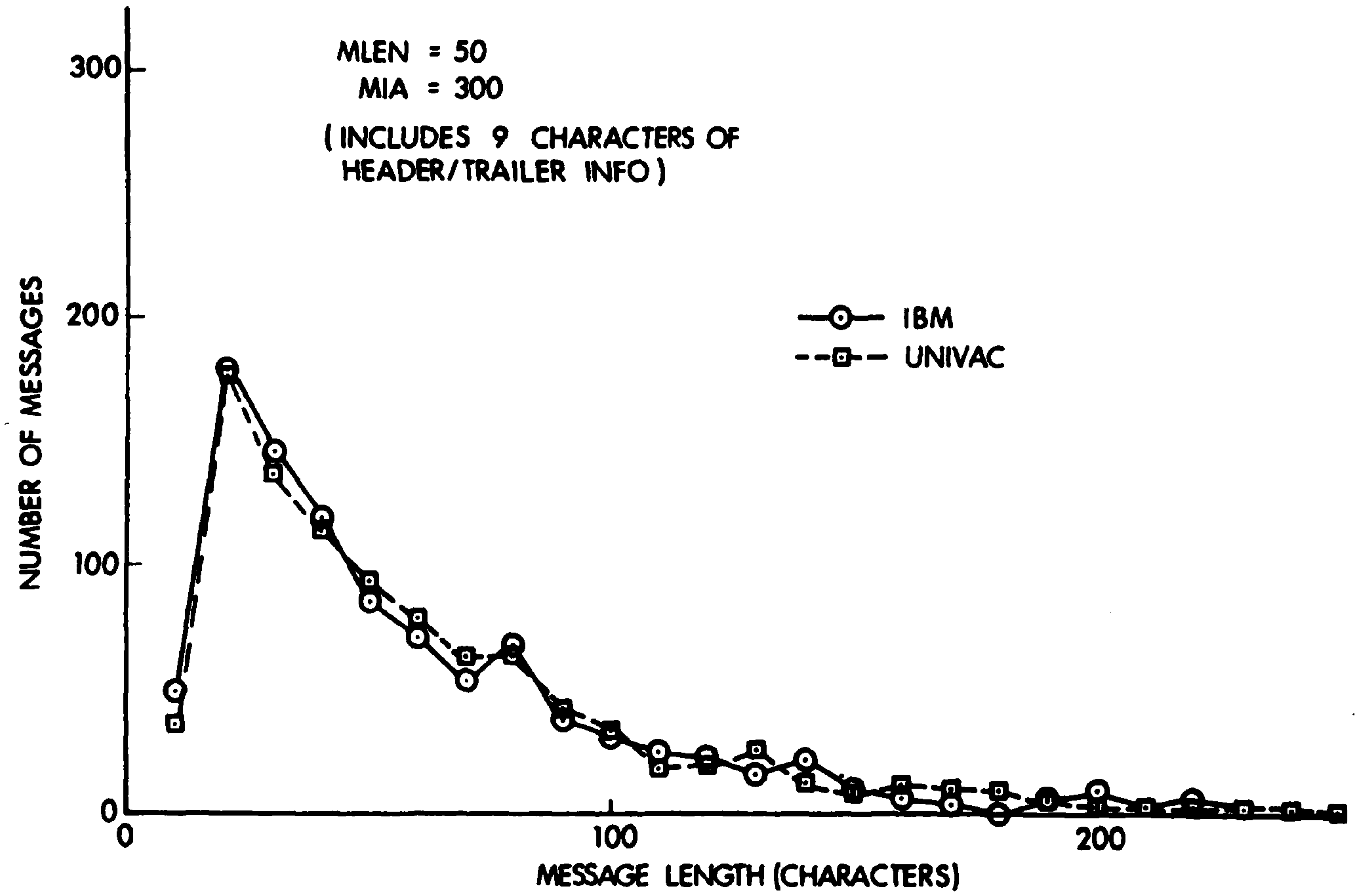


Figure 3.4. Message Length Distribution (DLCN).

The IBM plots are based on a sixty point user defined continuous approximation function, and the UNIVAC plots are based on the GPSS 1100 intrinsic exponential function.

3.6.2 Some Effects of Varying Pseudo Random Number Sequences. Plots of end to end delay (or total transmission time) versus message arrival rate at each node are shown in Figure 3.5; the two plots shown are for both IBM and UNIVAC simulations of the six node DLCN ring using different combinations of pseudo random number generators (or the same generator in the IBM case differently seeded). For runs of 1000 message terminations, the end to end delay is obviously sensitive to the sequences of pseudo random numbers used. To smooth these differences one can make several simulation runs using either different sets of random number generators or different sets of seeds and then either take the mean of the results associated with each designated interarrival time, or construct the final curve using minimum mean square error fit. This would be the case if fixed termination counts are used or if the simulation is stopped at a fixed time. A statistically better approach would be to design the stopping criteria to take cognizance of the confidence intervals involved with the statistics of the output data, as mentioned earlier.

3.6.3 Nominal Versus Measured Parameters. Differences were observed in UNIVAC and IBM GPSS outputs for DLCN simulations using identical nominal parameters for both mean message length and mean interarrival time. The differences in observed mean message lengths are essentially constant for all corresponding interarrival times (for UNIVAC a mean of 58.4$\pm$ 0.2 characters and for IBM a mean of 57.9 $\pm$ 0.1, making the worst case difference approximately 1% of nominal mean of 59 characters including the 9 character overhead). Because the differences in observed mean message length are essentially constant, only differences in mean interarrival times appear to be significant. For the six node DLCN simulation with a nominal mean message length of 50 characters (excluding overhead) two curves are shown in Figure 3.6 for both IBM and UNIVAC simulation results for total message transmission time (i.e., end to end delay). The curves marked "nominal" are plotted using the nominally specified nodal interarrival times. The curves marked "adjusted" use an abscissa of observed mean nodal interarrival times. The "total" time ordinates using nominal interarrival time values are skewed to the high side for the UNIVAC results and are skewed slightly to the low side for the IBM results, thereby giving a more pessimistic estimate of system performance for UNIVAC data and a more optimistic estimate of performance for IBM data than is the case if observed mean interarrival times are used as abscissas.

3.6.4 Results for Newhall Loop. Simulation results for total transmission times versus per node message arrival rate for the Newhall Loop Network are shown in Figure 3.7. Curves for IBM GPSS/360, UNIVAC GPSS 1100, and the published data of Reames and Liu [9] are shown for comparison. The differences are likely caused by variations in the actual pseudo random number sequences used in each case coupled with the 1000 transmitted messages stopping criterion. The IBM/360 and UNIVAC

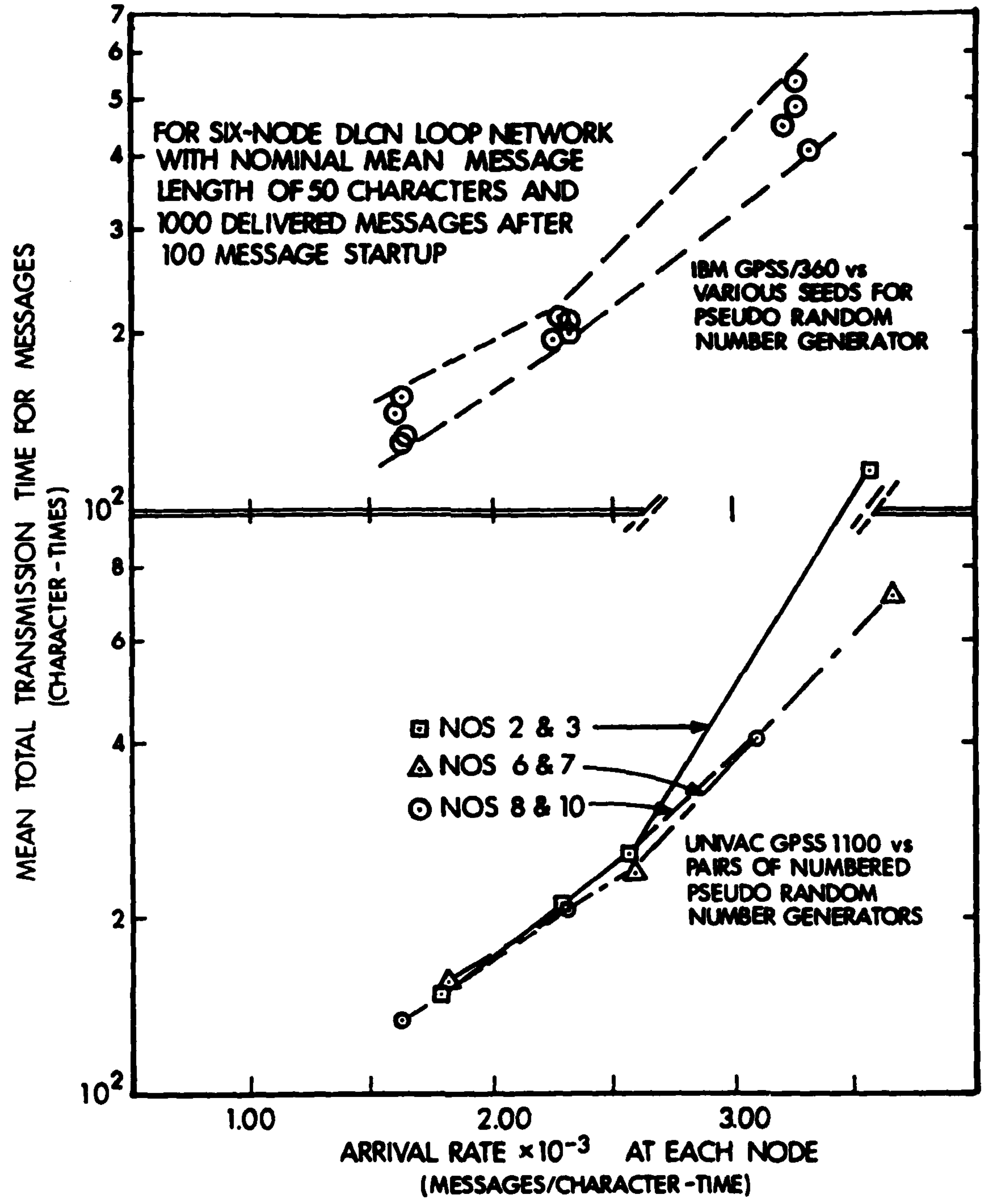


Figure 3.5. Total Transmission Time vs Message Arrival Rate at Each Node.

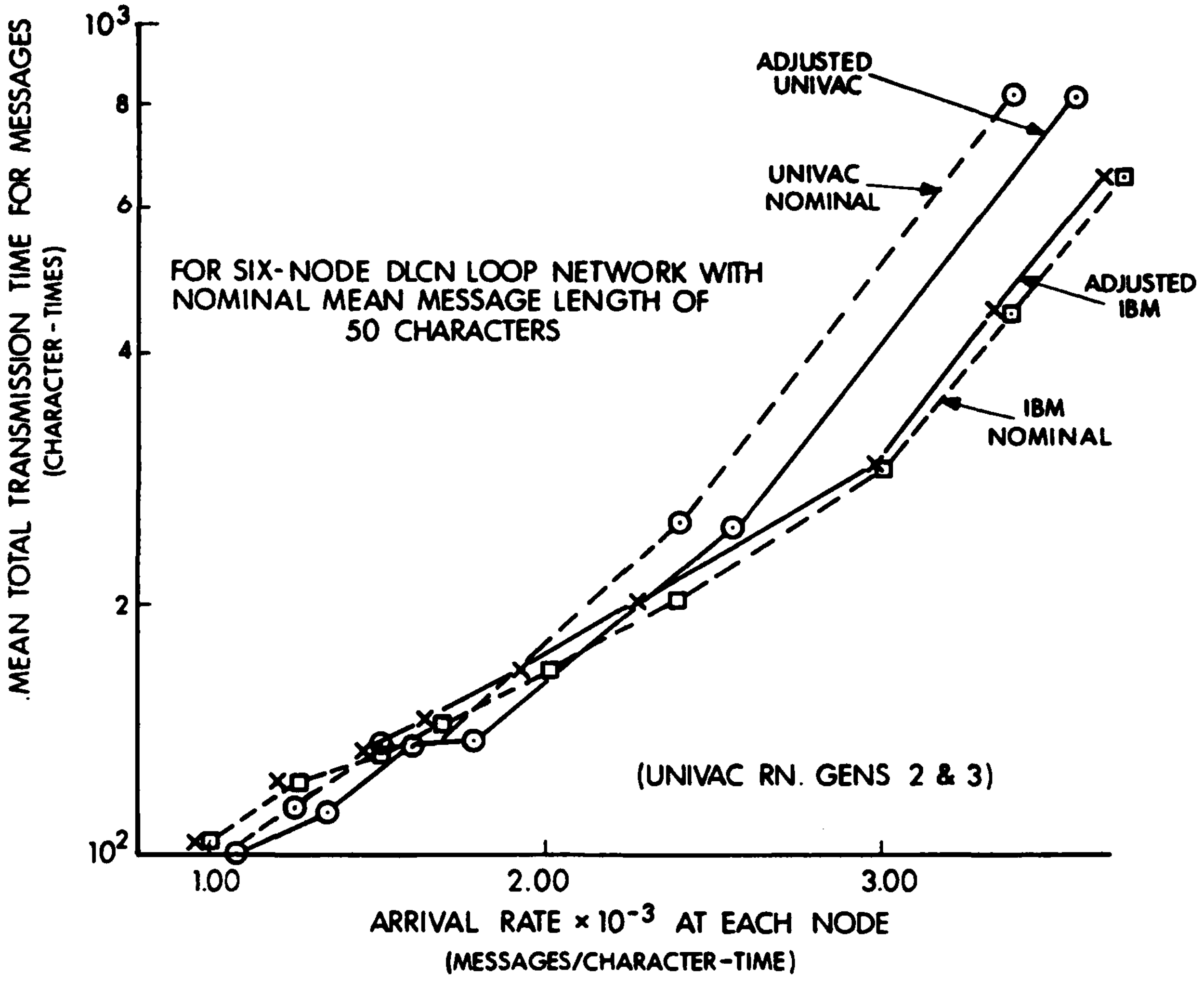


Figure 3.6. Total Transmission Time vs Message Arrival Rate at Each Node.

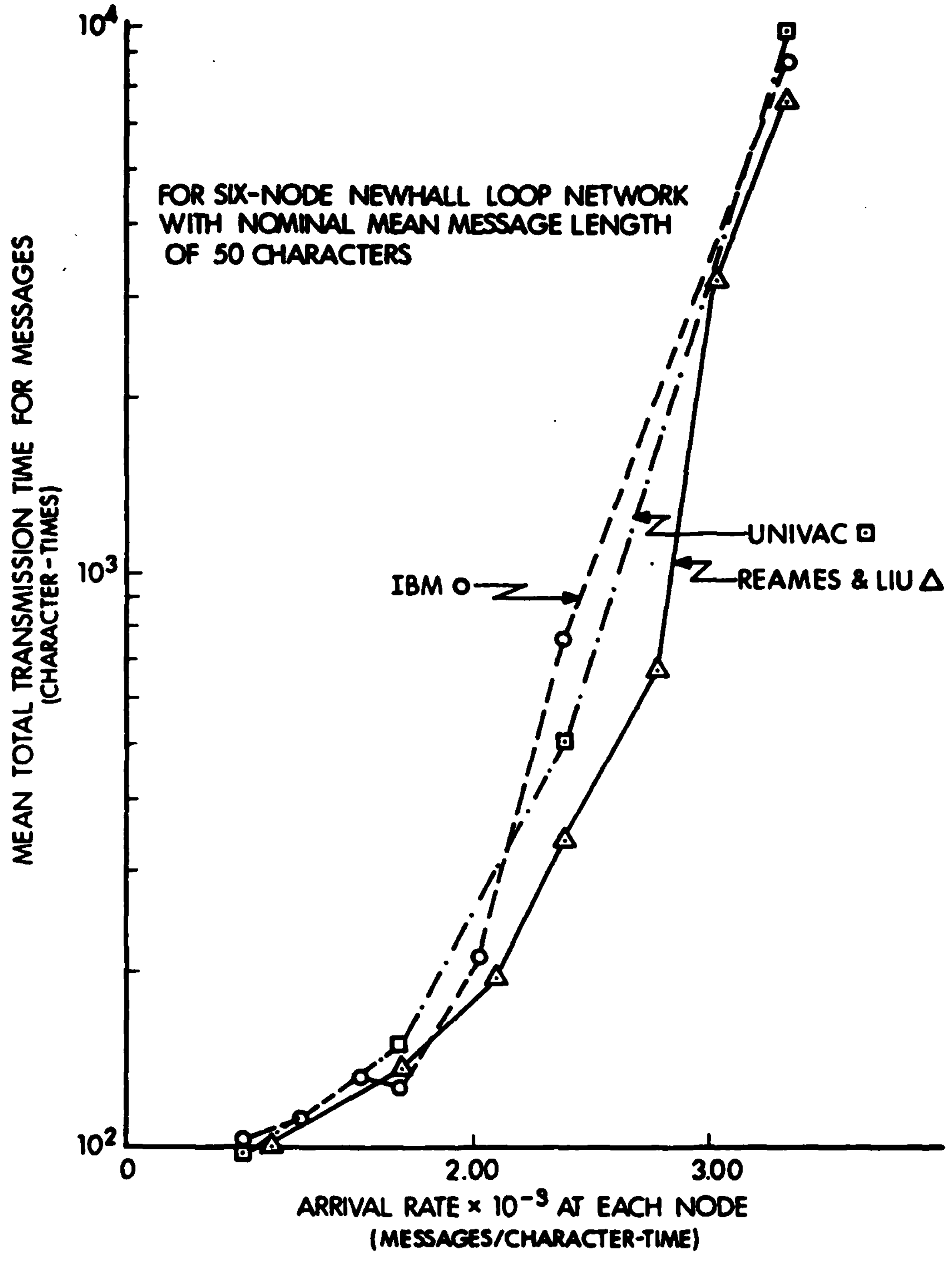


Figure 3.7 Total Transmission Time vs Message Arrival Rate at Each Node.

results match reasonably well, indicating that successful and correct translation between syntactically different dialects of GPSS is feasible.

3.6.5 Comparison of Results For All Four Networks. Paralleling the study reported in [9], the primary quantities of interest in this study are the mean total transmission time for messages (i.e., end to end delay) and mean queueing time for messages; however, many other quantities such as communication link utilizations were also measured in these simulations. Some of the relevant times that are discussed further are defined below:

(1) queueing time--time elapsed from message generation until placement on the loop by the transmitter at the source node;

(2) transmission time--time elapsed from message placement on the loop until the last character is received and removed from the loop at the destination;

(3) acknowledgement time--time elapsed from generation of the acknowledgement message at the destination node until the last character is received at the source node;

(4) total message transmission time (or end to end delay time)--sum of (1) and (2) only for Newhall and Pierce loops; sum of (1), (2) and (3) for DLCN (including DLCNNE); and modified sum of (1), (2) and (3) for Playthrough, where Playthrough's simulation differs from the others in that detailed simulation of character by character transmission does not take place; rather, [control message--data message--control message] groupings of characters are used for simulator efficiency, and the acknowledgement rides for free on the trailing control message. (Note that inclusion of character error simulations, not currently used in any of the loop network simulations, would likely require modification of Playthrough code to perform character by character transmission between transmitter-receiver pairs around the ring in a fashion similar to the other three simulation models. Such modification would also tend to increase the running time for the Playthrough simulation.)

The general characteristics of all four networks modeled are the same. Each comprises six nodes, with each message source being an identical and independently distributed Poisson process. Messages produced at each node are addressed uniformly to the other five nodes, so that message traffic is entirely symmetric and random. Message data lengths are assumed to be exponentially distributed with a nominal mean of 50 characters; actually, a truncated exponential distribution is used with no message exceeding 500 characters in length in order not to violate the hardware defined maximum length message including overhead of 512 characters that DLCN was assumed capable of handling. For the three loops other than Playthrough, nine additional characters of header information were added to each message or packet produced; the Playthrough simulation adds ten characters of overhead in the following way: three characters of control message information to initiate transmission on the loop, four characters of overhead added to the data message in the form of two characters of message length information and two characters for error detection, finally followed by three characters of control information to terminate the loop connection from source to destination and to carry acknowledgement information from destination to source. All timing is in arbitrary character-time units, so that no particular line rate is assumed. Propagation delay on the communication channel itself was ignored. In the three models other than Playthrough each ring interface unit through which messages pass contributes two units of delay: one unit in the receiver for address checking and one unit in the transmitter. In Playthrough GO is delayed by only one time unit in ring interfaces with nothing to transmit, and is delayed by three time units when preceded by a three character control message to allow time for address checking and control message transformation at appropriately designated nodes before relay by the ring interface transmitter. Special features in the DLCN model are described further in [9].

Tables 3.1 through 3.4 present relevant results of the simulations for the four ring networks under consideration. In all of these tables certain abbreviations are common and are discussed in this paragraph. More specific labels and names relating to measured quantities and names used in the program listings in Appendices B and C are discussed in corresponding specific subparagraphs below. The first column in each table lists the nominal mean message interarrival time at each node in the corresponding network. Again, the units are character-times. Because the network (or system) comprises six nodes, the nominal mean system interarrival time is one sixth of this value. The third and fourth columns display both mean and standard deviation of the measured system interarrival times as tabulated in the programs using the symbolic name MSGAR in the Newhall, Pierce, and Playthrough models, and the name GENAR in the DLCNNE model. The node arrival rate shown in column two is computed as the reciprocal of six times the mean system interarrival time value from column three. Columns five and six show means and standard deviations for the measured mean message lengths (with program name MSGLN). The reasons these values differ significantly from the nominal mean of fifty characters are due to both underestimation of target mean by the truncated exponential using the IBM pseudo random number generator and to the way in which header characters are accounted for, as discussed in the model specific paragraphs below. The seventh column lists mean facility utilization which is found by averaging the six facility mean utilizations. Each facility (or transmitter) utilization essentially measures utilization of the corresponding outgoing communication link.

3.6.5.1 Model Specific Items for Newhall. Table 3.1 displays means and standard deviations of two simulation output parameters of intense interest and a third of only moderate interest. The mean total queueing time experienced by all messages arriving for transmission anywhere in the system of six nodes is tabulated in the simulation model under the name TLQTM and is listed in Table 3.1 as one entry for each corresponding message interarrival time. Message transmit time is shown under the heading TRNTM, and total message transmission time which is approximately the sum of TLQTM and TRNTM (though tabulated separately in the model) is shown under the heading TMGTM. The measured mean message length tabulated under heading MSGLN is based on a nominal mean message length of 50 plus 9 overhead characters (or 59 characters).

3.6.5.2 Model Specific Items for Pierce. For the Pierce loop simulation results the measured mean message length is nearly the nominal mean value of 50 characters. The nine character overhead is added to each packet which consists of at most 36 characters, and the average number packets per message (NPKMG) is 2.35. The average packet synchronization time (SYNTM) is 17.4; the average packet transmit time (PTRTM) is 46.6, with standard deviations shown in Table 3.2. The columns labeled PKWTM display packet waiting time statistics, and under TPKTM display total packet transmit time. The main parameter of interest is the total message transmission time displayed under TMGTM.

3.6.5.3 Model Specific Items for DLCNNE. Measured mean message length for the DLCN simulation with character error generation facilities disabled as shown in Table 3.3 is based on a nominal mean length of 50 characters plus nine characters of overhead. Means and standard deviations for the following parameters of interest are displayed in the remaining columns of Table 3.3. Statistics for total queueing time are shown under heading TRQTM; those for total transmit time for data messages on the way to their destinations is shown under RCVTM, and total transmit time for the return acknowledgement message is shown under ACKTM. TLATM is the total message transmission time which is (approximately) the sum of TRQTM, RCVTM, and ACKTM, and it is this value that is plotted in Figure 3.8. DLYTM records statistics for the per node time messages spend in delay buffers enroute to their destinations.

3.6.5.4 Model Specific Items for Playthrough. Measured mean message lengths shown in Table 3.4 for the Playthrough model are based on a nominal mean message length of 50 plus 4 overhead characters for a total of 54 characters. The six additional control message characters needed to start and stop data message transmissions affect queueing and total time statistics, but are not included in the message length statistics. Only the parameters of greatest interest are shown in Table 3.4, namely total queueing time under heading TLQTM and total transmission time (plotted in Figure 3.8) shown under heading TTLTM. TTLTM includes the acknowledgement time embedded in the control mechanism. (Note, message transit time is the difference: TTLTM minus TLQTM.) Table 3.5 displays additional information for the Playthrough loop, where mean queueing times versus distance (in number of nodes to the destination) are tabulated. Average waiting times for messages with destinations one hop away are shown under heading TLQ1, and those for messages with destinations five nodes away are shown under heading TLQ5. The maximum number of messages waiting in any of the six queues as well as the average number of messages waiting in queue during the simulation are also tabulated against corresponding message interarrival times per node.

TABLE 3.1

NEWHALL LOOP SIMULATION RESULTS
NOMINAL MEAN MESSAGE LENGTH = 50 CHARACTERS
TIME UNITS ARE "CHARACTER TIMES"

NOMINAL IAT/NODE	NODE ARRIVAL RATE X10^{-3}	SYSTEM INTER-ARRIVAL TIME MSGAR		MEASURED MSGLN		MEAN FACILITY UTILI-ZATION	TLQTM		TRNTM		TMGTM	
		MEAN	STD	MEAN	STD		MEAN	STD	MEAN	STD	MEAN	STD
1000	0.97	172.3	164.6	57.3	52.1	0.22	41.1	89.4	62.5	52.4	103.6	104.7
800	1.17	142.8	140.3	57.8	48.4	0.25	51.0	102.7	62.7	48.4	113.7	114.8
667	1.40	119.0	116.9	57.8	48.4	0.28	69.9	148.8	62.7	48.6	132.6	157.6
600	1.66	100.6	104.6	58.7	47.2	0.32	64.2	86.0	63.6	47.1	127.9	96.7
500	1.93	86.2	82.5	57.3	52.1	0.36	152.2	241.8	62.1	52.2	214.8	248.6
420	2.39	69.8	69.9	59.1	51.1	0.44	712.2	968.0	64.4	51.5	777.2	968.0
300	3.28	50.8	51.9	59.8	51.5	0.50	7830.4	4576.0	63.2	47.0	7894.9	4576.0
270	3.69	45.2	44.3	57.9	53.4	0.48	12978.7	6272.0	64.5	54.6	13029.2	6272.0

TABLE 3.2

PIERCE LOOP SIMULATION RESULTS
NOMINAL MEAN MESSAGE LENGTH = 50 CHARACTERS
PACKET LENGTH = 36 CHARACTERS
TIME UNITS ARE "CHARACTER TIMES"

NOMINAL IAT/NODE	NODE ARRIVAL RATE $X10^{-3}$	SYSTEM INTER ARRIVAL TIME MSGAR		MEASURED MSGLN		MEAN FACILITY UTILI-ZATION	NPKMG	SYNTM	PKWTM		PTRTM		TPKTM		TMGTM	
		MEAN	STD	MEAN	STD		MEAN	MEAN	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD
1500	0.66	253.3	241.1	47.9	51.4	0.164	2.32	18.0	25.3	88.3	46.0	16.1	71.3	90.3	126.7	110.6
1000	0.99	158.6	166.6	48.0	51.6	0.251	2.32	17.4	40.9	95.2	46.8	16.1	87.8	95.9	140.6	127.9
800	1.21	138.0	138.4	49.4	48.5	0.315	2.37	17.6	45.8	16.3	47.1	16.3	93.0	98.6	152.1	124.8
600	1.65	101.1	96.4	48.2	51.6	0.419	2.33	17.6	97.8	183.3	46.9	16.1	144.5	184.0	193.3	198.6
500	1.94	85.9	86.6	49.5	48.5	0.497	2.37	16.9	113.3	205.1	46.4	16.3	159.7	206.6	213.7	226.0
420	2.35	70.9	67.7	48.3	51.7	0.587	2.33	17.2	183.6	264.0	46.2	16.1	229.5	263.0	283.1	288.0
300	3.26	51.1	51.5	49.4	48.4	0.838	2.37	17.7	567.5	617.0	46.2	16.1	612.7	616.0	679.1	637.0
270	3.63	46.0	44.1	48.2	50.1	0.912	2.33	17.7	1719.4	1670.0	46.7	16.6	1766.5	1671.0	1804.6	1682.0

TABLE 3.3

DLCNNE (NO TRANSMISSION ERRORS) LOOP SIMULATION RESULTS
NOMINAL MEAN MESSAGE LENGTH = 50 CHARACTERS
TIME UNITS ARE "CHARACTER TIMES"

NOMINAL IAT/NODE	NODE ARRIVAL RATE X10^{-3}	SYSTEM INTER-ARRIVAL TIME GENAR		MEASURED MSGLN		MEAN FACILITY UTILI-ZATION	TRQTM		RCVTM		ACKTM		TLATM		DLYTM	
		MEAN	STD	MEAN	STD		MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD
1500	0.65	256.6	274.4	57.9	52.9	0.132	6.4	27.8	63.9	62.2	18.8	29.3	89.1	78.1	13.0	36.2
1000	0.97	171.0	164.8	57.9	52.9	0.205	10.7	36.8	70.5	77.9	22.3	37.0	103.6	100.6	15.0	40.2
800	1.22	136.8	131.9	57.9	52.9	0.248	14.2	46.3	72.9	80.9	27.5	54.9	114.6	114.5	16.5	42.8
667	1.46	114.0	109.9	57.9	52.9	0.297	18.9	62.1	77.1	88.2	32.3	59.0	128.2	126.6	18.3	45.9
600	1.62	102.6	98.9	57.9	52.9	0.349	26.1	70.3	86.2	95.2	38.2	68.1	150.4	146.9	21.3	49.4
500	1.95	85.5	82.4	57.9	52.9	0.429	36.1	99.5	100.2	131.0	47.0	74.2	183.3	196.9	25.9	61.5
420	2.32	71.8	69.3	57.9	52.9	0.522	47.7	124.1	126.1	171.3	57.4	84.1	231.2	240.4	33.1	73.7
333	2.93	56.9	54.9	57.9	52.9	0.662	105.6	346.0	196.2	360.0	88.2	117.7	389.8	574.0	53.3	132.8
300	3.26	51.2	49.3	57.9	52.9	0.757	124.4	385.0	257.2	499.0	113.7	157.4	494.7	728.0	70.4	184.9
270	3.63	45.9	44.3	57.8	52.8	0.849	231.9	479.0	451.1	856.0	151.8	176.1	836.8	1067.0	117.1	313.0

TABLE 3.4

PLAYTHROUGH LOOP SIMULATION RESULTS
NOMINAL MEAN MESSAGE LENGTH = 50 CHARACTERS
TIME UNITS ARE "CHARACTER TIMES"

NOMINAL IAT/NODE	NODE ARRIVAL RATE X10^{-3}	SYSTEM INTER-ARRIVAL TIME MSGAR		MEASURED MSGLN		MEAN FACILITY UTILI-ZATION	TLQTM		TTLTM	
		MEAN	STD	MEAN	STD		MEAN	STD	MEAN	STD
1500	0.65	256.6	247.8	52.9	52.9	0.273	27.0	75.1	113.8	100.0
1000	0.97	171.0	165.1	52.9	52.9	0.347	41.6	92.6	128.9	115.7
800	1.22	136.8	132.1	52.9	52.9	0.380	58.3	121.9	145.9	142.5
667	1.46	114.0	110.4	52.9	52.9	0.437	76.8	145.1	164.8	162.3
600	1.62	102.6	99.2	52.9	52.9	0.441	87.1	158.9	175.2	175.1
500	1.95	85.5	82.7	52.9	52.9	0.490	107.2	178.4	196.0	191.5
420	2.32	71.8	69.4	52.9	52.9	0.570	197.0	276.0	286.3	286.0
333	2.93	56.8	55.1	52.8	52.8	0.663	449.2	601.0	540.1	602.0
300	3.26	51.1	49.3	52.8	52.7	0.730	1072.7	1260.0	1163.1	1260.0
270	3.60	46.3	44.6	52.1	52.1	0.756	2198.5	2671.0	2277.7	2662.0

TABLE 3.5

PLAYTHROUGH LOOP SIMULATION RESULTS
MEAN QUEUEING TIME VERSUS DISTANCE TO DESTINATION
NOMINAL MEAN MESSAGE LENGTH = 50 CHARACTERS
TIME UNITS ARE "CHARACTER TIME"

NOMINAL IAT/NODE	MEAN QUEUEING TIME VS. DISTANCE					MAX NO. MSGS IN ANY QUEUE	AVERAGE QUEUE CONTENTS (IN MESSAGES)
	TLQ1	TLQ2	TLQ3	TLQ4	TLQ5		
1500	14.9	18.8	31.3	33.3	39.5	3	0.017
1000	28.4	33.3	43.3	43.1	58.1	3	0.040
800	33.2	42.6	64.2	66.6	82.5	3	0.071
667	44.6	47.9	85.7	86.8	117.0	4	0.112
600	45.2	63.0	93.3	106.5	128.3	4	0.140
500	66.4	77.0	114.4	142.6	139.6	5	0.208
420	81.7	129.3	225.5	258.8	283.3	5	0.457
333	126.2	255.1	436.7	614.1	848.9	9	1.315
300	152.6	478.8	852.6	1830.2	2110.9	13	3.510
270	203.8	657.1	1982.7	3620.7	4924.8	27	8.340

3.7 Findings.

The data generated for Newhall, Pierce and DLCNNE loops agree reasonably well with published data [9] in that the relative positions of the plotted total transmission time data are similar. The exact values differ somewhat, which for Newhall and Pierce can be accounted for by pseudo random number generator variations. DLCNNE differs from DLCN results because of the disabling of the erroneous message generation and retransmission scheme resulting in an approximately ten per cent difference in computed values as discussed in Section 3.5.2 of this report.

The significance of Figure 3.8 is that it provides the first extensive comparison between the DLCN and Playthrough link level protocol schemes. Overall transmission times for DLCN are lower on the average than for the other link level protocol schemes, and this is to be expected. Under heavy loading, the Newhall, Playthrough, and even Pierce schemes suffer from increased queueing delays, whereas the DLCN scheme is designed to minimize queueing delays. Nothing is free, however,and in the DLCN scheme messages suffer random exponentially increasing delays en route to their destinations, making strict timeouts for error control difficult. Transit times in Playthrough grow approximately linearly with almost imperceptible slope, so that as in Newhall, once a message transmission is initiated it proceeds rapidly and is completed in almost fixed time. The disadvantage of Playthrough is that under heavy load, queueing time grows exponentially because long hop messages must wait long times before a sufficient number of links from source to destination nodes become simultaneously free.

These disadvantages are common among schemes that use dedicated circuit switching in the transmission of messages. Packet switched schemes tend to experience less rapid growth in queueing time under heavy loads; however, they require dedicated intelligence or capacity in either the ring interface processor or in the attached component (e.g., the host computer) to packetize messages at their sources and to reassemble at their destinations packets that are arriving in arbitrary sequence from possibly disparate messages. DLCN employs variable length packets in this simulation up to a maximum of 512 characters in length, which represents a chosen hardware limit. Messages of longer length were not allowed in this simulation because the code to packetize them was not included in the model. DLCN minimizes queueing times by usually placing on the ring newly arriving messages ahead of messages already on the ring through the use of expandable delay buffers. This technique appears to be a particularly effective means of maintaining reasonable mean transmission times under heavier loads than is possible with the other link level schemes. An advantage of the Newhall and Playthrough protocols is their ability to transmit quickly messages of any arbitrary length when the number of characters arriving for transmission to the entire network does not exceed the burst character transmission rate.

An interesting observation from examining the plots in Figure 3.8 is that the perpetually circulating control token in the Playthrough

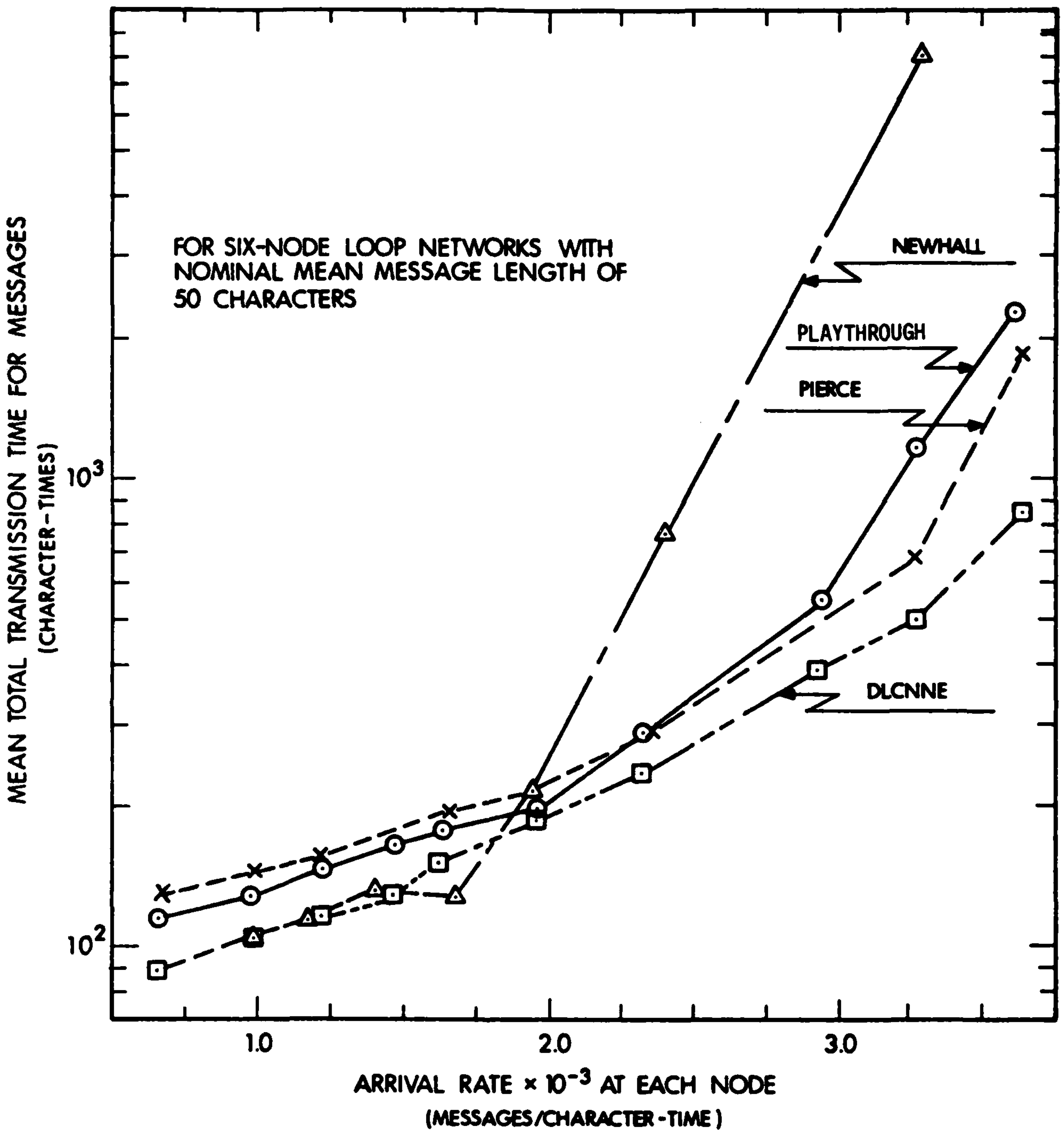


Figure 3.8 Total Transmission Time vs Message Arrival Rate at Each Node.

scheme tends to have a packetizing effect on mean total transmission times; so that for non-saturating loads it corresponds to but is lower than the mean total transmission time for the Pierce scheme.

Neither the Pierce nor the Newhall simulations include the loading effects and delays produced by the inclusion of acknowledgements for messages sent; whereas, Playthrough and DLCNNE do include them. The simulation results are therefore favorably biased for Pierce and Newhall.

4. CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER WORK

4.1 GPSS Capabilities.

Some capabilities of the GPSS language for modeling and simulating systems were presented in Chapter 2, and differences in two available implementations of this language were discussed. GPSS has several facilities for system level modeling of computer communication networks. Messages are easily modeled as dynamic entities, called transactions. Language features are provided for generating or implementing message arrivals and other randomly occuring events such as link and node failures and dynamic routing scheme choices. Equipment entities such as transmitters, receivers and message queues are easily modeled. Both automatic and user specified means for collecting and computing statistics for message transmission such as mean, variance, and distributions (percentiles) of queueing, transmission, and end to end delay times are also included. These statistics can be used to predict system behavior under varying conditions.

4.2 Sample Simulation Results and Applications.

To illustrate use of these capabilities, GPSS models of several ring topology computer communication networks were examined in Chapter 3. Data were collected to indicate performance under varying system load for each of the ring network link level protocols considered. These performance data were plotted to show relative performance of the differing link control and message handling schemes. Tests for statistical validity of these data should be performed before decisive conclusions are drawn from these comparisons. It was the purpose of this study to demonstrate use of GPSS for computer communication network modeling rather than to produce statistically valid system comparisons. However, some statistical validity tests for the Playthrough data were performed using the method of batch means employed by Wolf [21] in his simulation of a double loop DLCN configuration. For instance, the mean total transmission time entries (TTLTM) in Table 3.4 satisfy a 90% confidence level test at nominal interarrival times of 300, 600, and 1000. This suggests reasonable accuracy in the plotted performance data.

The simulations that have been run have used a nominal mean message length of 50 characters (some messages are longer and some are shorter). This mean message length approximates the characteristics of many actual computer communication schemes, but the actual distributions involved as well as their means may vary somewhat from this choice. To gain an appreciation of how well the ring network schemes considered here might work in a typical computer communications structure, we must make additional assumptions about mean interarrival times for messages, the number of binary digits or bits used to encode the characters,and the speed of the communication links in the network in bits per second to make it independent of modulation scheme.

In order to use the data presented in Chapter 3 recall we have assumed that messages are on the average 50 and no more than 500 characters in length with length governed by a truncated exponential distribution. If one assumes 10 bits are required to transmit one character (7 code bits,1 parity bit, 1 start bit, and 1 stop bit for an asynchronous format), then one "character time" at a link transmitter/receiver speed of 1 million bits per second (1 Mbps) is 10^{-5} seconds, and at a speed of 1200 bps is 8.33×10^{-3} seconds. Assuming a network of six identical data terminals in which operators send messages to some destination node at an average rate of one every 30 seconds, then we compute the communications network (i.e., system) arrival rate as: multiply six (the number of nodes corresponding to the simulation results presented) times the per node arrival rate (in messages per second) times the time for one character (in seconds per character time). At a line speed of 1 Mbps these assumptions result in a mean system arrival rate of 0.002×10^{-3} messages per character time (or 0.00033×10^{-3} messages per character time per node). Looking in Figure 3.8 under this arrival rate, one finds that for all four ring network structures considered the expected mean total transmission time for messages is less than 200 character times which corresponds to less than 2 milliseconds. At a link speed of 1200 bps using otherwise same assumptions, the per node arrival rate is 2.78×10^{-3} messages per character time. At this arrival rate Figure 3.8 says that for all but the Newhall scheme the expected message transmission time is less than 540 character times (or 4.5 seconds); for the Newhall scheme the expected message transmission time is approximately 4000 character times or 33 seconds, not a very desirable performance if one expects to generate a new message for transmission once every 30 seconds.

4.3 Simulation Language Alternatives.

Having the capability to simulate various computer communication networks quickly permits analysts to identify potential bottlenecks and deficiencies in proposed computer communication network schemes. Various discrete event languages are available to facilitate the programming of these simulation models. Two of the more popular are various dialects of GPSS and SIMSCRIPT. GPSS is a block oriented language in which simulator specifications relate more to the flow of dynamic entities in the actual model than to traditional computer programming languages. GPSS is interpreted rather than compiled as is SIMSCRIPT. Various comparisons of these languages [23] and [24] point to advantages and disadvantages of each. Beginners usually have an easier time learning GPSS because of the abundance of tutorial material available; whereas, far less complete tutorial material is available to beginners learning SIMSCRIPT. Because SIMSCRIPT is compiled, some models written in this language can be expected to execute more rapidly than do similar models written in GPSS. Recent additions to the SIMSCRIPT language, however, tend to reduce its speed advantage [23]. Certain computations are more easily specified in one language than in the other. For instance, exponentiation is not available as a primitive and is cumbersome to specify in GPSS [21] p.111.

4.4 Use of GPSS.

Two dialects of GPSS (namely, GPSS/360 and GPSS 1100) were used in the example computer communication network simulations documented here. Differences in both syntax and semantics between the two dialects have been identified and are discussed in Chapter 2. Because of these differences, care should be exercised when comparing output data from one dialect with that from another in order to insure that the comparison is meaningful. It is however possible to correctly translate a model from one dialect to another by carefully tracing the flow of transactions in the two models to identify and correct or at least account for differences in interpreter execution (i.e., semantics). This task, however, is not particularly easy and should not be taken lightly.

Because of the variety of programming techniques required to implement the several ring network simulation models in GPSS, the collection of programs found in appendices B and C coupled with those in [6] should be a valuable aid to programmers seeking to model other computer communication network architectures and protocols. Each protocol considered has its own peculiar implementation requirements that relate to other actual and potential computer network structures.

4.5 Future Work.

Because of recognized deficiencies in the GPSS language, such as long execution times and cumbersome constructions to do simple computations directly available in other languages, an investigation into the use of the discrete event simulation language SIMSCRIPT II.5 should be considered for further work. There are indications that SIMSCRIPT II.5 is superior to GPSS because of its generally higher speed of execution and lower memory space requirements for the same model [23] and [24]; also, model implementation reportedly requires programmer skill roughly equivalent to that of a competent FORTRAN or ALGOL programmer, which should cause little difficulty for most organizations. The set of examples considered in Chapter 3 could be used as a starting point(and validation check) for initial SIMSCRIPT II.5 modeling efforts. Use of SIMSCRIPT will not necessarily replace the use of GPSS because some investigators [24] indicate that it is likely to be faster to program an initial system model in GPSS to get quick results that can be used to guide the development of a more comprehensive (and possibly more efficient) SIMSCRIPT II.5 model.

Because use of a ring network architecture has been proposed for SIGMA [29], the simulation models examined here should be considered for potential further use in evaluation of the SIGMA computer communications structure.

REFERENCES

1. T. J. Schriber, Simulation Using GPSS, John Wiley & Sons, New York, NY, 1974.

2. International Business Machines Corp., General Purpose Simulation System/360 User's Manual, H20-0326-2, White Plains, NY, 1968.

3. Sperry Rand Corp., UNIVAC 1100 Series General Purpose Systems Simulator (GPSS 1100), Programmer Reference UP-7883 Rev. 1, Blue Bell, PA, 1974.

4. J. W. Wong, "Queueing Network Modeling of Computer Communication Networks," Computing Surveys, Vol. 10, No. 3, September 1978, pp. 343-351.

5. F. Basket, K. Chandy, R. Muntz, and F. Palacios, "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers," J. ACM, Vol. 22, No. 2, April 1975, pp. 248-260.

6. C. C. Reames, "System Design of the Distributed Loop Computer Network," PhD dissertation, Department of Computer and Information Science, The Ohio State University, Columbus, OH, March 1976; available from University Microfilms International, Ann Arbor, MI, Order No. 76-18, 028.

7. M. J. Sobel, "Simulation and Evaluation of the 'Play Through' Protocol for Distributed Control of Ring Networks," M.S. Thesis, Electrical Engineering Department, University of Maryland, College Park, MD, May 1980.

8. C. C. Reames and M. T. Liu, "A Loop Network for Simultaneous Transmission of Variable Length Messages," Proceedings 2nd Annual Symposium On Computer Architecture, Houston, TX, January 1975, pp. 7-12.

9. C. C. Reames amd M. T. Liu, "Design and Simulation of the Distributed Loop Computer Network (DLCN)," Proceedings 3rd Annual Symposium on Computer Architecture, Clearwater, FL, January 1976, pp. 124-129.

10. T. C. Wilson and C. B. Silio, Jr., "Loop Interconnection of Processors Using a 'Play Through' Protocol," Proceedings 9th International Symposium on Mini and Micro Computers" (MIMI '79), Montreal, Quebec, Canada, September 1979, pp. 175-180.

11. T. C. Wilson and C. B. Silio, Jr., "Extensions to the 'Play Through' Protocol for Loop Networks", Proceedings IEEE Workshop On Interconnection Networks for Parallel and Distributed Processing," W. Lafayette, IN, April 1980, pp. 103-104.

12. T. C. Wilson and C. B. Silio, Jr., "Distributed Control of Ring Networks Using a 'Play Through' Protocol," Proceedings 21st IEEE Computer Society International Conference (COMPCON Fall '80), Washington, DC, September 1980, pp. 507-515.

REFERENCES (Continued)

13. W. D. Farmer and E. E. Newhall, "An Experimental Distributed Switching System to Handle Bursty Computer Traffic," Proceedings of the ACM Symposium on Problems in the Optimization of Data Communication Systems, Pine Mountain, GA, October 1969, pp. 1-33.

14. R. A. Connan and J. R. Kersey, "Synchronous Data Link Control: A Perspective," IBM Systems Journal, Vol. 13, No. 2, 1974, pp. 140-162.

15. D. J. Farber, and K. C. Larson, "The System Architecture of the Distributed Computer System - The Communications System," Proceedings of Symposium on Computer-Communications Network and Teletraffic, Polytechnic Institute of Brooklyn, NY, April 1972, pp. 21-27.

16. J. R. Pierce, C. H. Coker, and W. J. Kropfl, "An Experiment in Addressed Block Data Transmission Around a Loop," IEEE International Convention Record, March 1971, pp. 222-223.

17. J. R. Pierce, "Network for Block Switching of Data," Bell System Technical Journal, Vol. 51, No. 6, July-August 1972, pp. 1133-1145.

18. J. R. Pierce."How Far Can Data Loops Go?," IEEE Trans. on Communications, Vol. COM-20, June 1972, pp. 527-530.

19. R. R. Anderson, J. F. Hayes, and D. N. Sherman, "Simulated Performance of a Ring Switched Data Network," IEEE Trans. on Communications, Vol. COM-20, No. 3, June 1972, pp. 576-591.

20. H. Jafari, J. Spragins, and T. Lewis, "A New Modular Architecture for Distributed Computer Systems," IEEE Proc. of Syposium on Distributed Processing: Trends and Applications, Gaithersburg, MD, June 1978, pp. 72-77.

21. J. J. Wolf, "Design and Analysis of the Distributed Double-Loop Computer Network (DDLCN)," PhD dissertation, The Ohio State Univ., 1979.

22. International Business Machines Corp., GPSS/360 System Manual, GV20-0075-0, White Plains, NY, 1967.

23. J. M. Scher, "Structural and Performance Comparisons Between SIMSCRIPT II.5 and GPSS V," Proc. of 9th Annual Pittsburgh Conf. on Modeling and Simulation, University of Pittsburgh, Pittsburgh, PA, April 1978, pp. 1267-1272.

24. I. M. Kay, T. M. Kisko, and D. E. Van Houweling, "GPSS/SIMSCRIPT--The Dominant Simulation Languages," Proc. 8th Annual Simulation Symposium, Tampa, FL, 1975, pp. 141-154.

25. D. E. Knuth, The Art of Computer Programming, Vol 2; Seminumerical Algorithms, Addison-Wesley, Reading, MA, 1969, pp. 34-66.

REFRENCES (Continued)

26. J. D. Gibbons, Nonparametric Methods for Quantitative Analysis, Holt, Rinehart and Winston, New York, NY, 1976, pp. 363-378.

27. J. E. Freund, Mathematical Statistics, 2nd. Ed., Prentice-Hall, Englewood Cliffs, NJ, 1971, pp. 352-355.

28. G. S. Fishman, Principles of Discrete Event Simulation, John Wiley & Sons, New York, NY, 1978, pp. 373-376.

29. OPTADS, "Draft System Specification for the Force Level Maneuver Control System (SIGMA)," PM OPTADS, Ft. Monmouth, NJ, March 1981.

Next page is blank.

APPENDIX A

ON THE RANDOMNESS OF PSEUDO RANDOM NUMBER GENERATORS USED IN IBM GPSS/360 AND UNIVAC GPSS 1100 LANGUAGES

Next page is blank.

APPENDIX A

ON THE RANDOMNESS OF PSEUDO RANDOM NUMBER GENERATORS USED IN IBM GPSS/360 AND UNIVAC GPSS 1100 LANGUAGES

A.1 INTRODUCTION

Tests of randomness were performed on the UNIVAC GPSS 1100 and IBM GPSS/360 pseudo random number generators when simulation models translated from one language to the other failed to yield comparable statistics for checkout runs. Initially, the translations themselves were suspect; however, subsequent investigation found no basis for faulting the translations.

The simulation models tested rely on pseudo random number generators embedded in the languages to generate message traffic for input to the models. Small differences in mean message lengths and mean interarrival times for this traffic were observed for corresponding runs in the two languages, and it was conjectured that these differences might be caused by nonrandom behavior in the underlying pseudo random number generators. Testing of the pseudo random number generators was thus begun. It is conjectured that if the generators cannot be rejected for nonrandom behavior using a set of standard statistical tests for randomness, then semantic differences in the implementation, instantiation, and/or interpretation of these two versions of GPSS are likely. Additional tests for these semantic differences are reported elsewhere.

The following sections provide a discussion of the testing of the generators, and the results of those tests.

A.2 TESTS SELECTED

A.2.1 Introduction to Randomness Tests.

Three standard tests of randomness were chosen in this study, namely: (1) the runs above and below the median test, (2) the maximum of five test and (3) the runs up and down test. Each of these tests attempts to determine if a generated sequence of numbers is sufficiently random by detecting either cyclical patterns or otherwise nonrandom behavior. All of the chosen tests are empirical in that a computer manipulates groups of numbers from the sequence and computes certain statistics which are compared with standard statistical tables[25]. While it is recognized that there are a great many randomness tests, these particular tests were chosen both because of their reputed reliability and the ease with which their algorithms could be adapted to a computer program[25]. Also, runs tests are perhaps the only statistical tests which focus on the order in sequence[26].

A.2.2 Runs Above and Below the Median Test.

The first test chosen was the runs above and below the median test. In this test a run is defined as a series of either numbers (or in the nonparametric approach, ranks) within the sequence having values strictly above or strictly below the value of the median observation. The nonparametric test method merely requires an ordered set of ranks, that is, the relative positions of the values of the observations within the sequence. Order is important because this test is based on runs.

A test statistic, i.e., a random variable whose values are determined by sample data [27], can be calculated based on the total number of runs in a sequence. This statistic may reveal nonrandom behavior in that either too few runs or too many runs would likely be the result of a trendy or cyclical pattern. The sampling distribution of the number of runs can be approximated by a normal distribution; therefore, a normal test is applied to the actual number of runs in the sequence [27].

The test statistic Z is defined as follows:

$$Z = \frac{u - E(u)}{[\mathrm{var}(u)]^{1/2}},$$

where u = number of runs in the sequence,

$$E(u) = \frac{2n_1n_2}{n_1 + n_2} + 1,$$

$$\mathrm{var}(u) = \frac{2n_1n_2\,(2n_1n_2 - n_1 - n_2)}{(n_1 + n_2)^2\,(n_1 + n_2 - 1)},$$

n_1= number of observations above the median, and

n_2 = number of observations below the median.

The test statistic Z is then compared to critical values obtained for a two-tailed normal test from which the critical region (the region where the hypothesis of randomness must be rejected) is determined. A two-tailed normal test assumes a normal distribution about some mean, and then a critical region is obtained for both the upper and lower tails of the distribution. If Z falls within the critical region, then the sequence is suspect and the generator for the sequence is dismissed as being nonrandom. A negative value of Z falling in the rejection region implies that there are not enough runs in the sequence; on the other hand, a positive value of Z falling in the rejection region is indicative of too many runs and possibly a repetitious pattern [27].

A.2.3 Maximum of Five Test.

The second test chosen was the maximum of five test. Knuth [25] points out that the use of this test for a moderately sized sequence will tend to detect both local and global nonrandom behavior. Local nonrandom behavior could likely be the result of clustering of observations around a single value while global nonrandom behavior might be due to the multiplier for the generator not being large enough (e.g., see Section A.4).

This test consists of obtaining observations U_{5j}, $U_{5j+1}, \ldots,$ U_{5j+4} for $j = 0, \ldots, m - 1$ where m is the integer quotient of n divided by 5, n being the total number of observations; let V_j be the maximum of each of these sequences of five numbers. The Kolmogorov-Smirnov (KS) test method for measuring the amount of deviation between an assumed distribution function and the empirical or actual distribution function is used here. The KS test is applied to the sequence $V_0, \ldots, V_{m-1}$, which is assumed to have the cumulative distribution function $F(x) = x^5$ $(0<x<1)$. It can be shown that the distribution function for the V_j's is indeed $F(x)$ [25]. The Kolmogorov-Smirnov test statistics K^+m and K^-m are then compared to standard statistical tables to determine whether the values lie within the critical regions for given confidence levels, where K^+m is the greatest amount of deviation when the actual distribution function is greater than $F(x)$, and K^-m is the greatest amount of deviation when the actual distribution function is less than $F(x)$. If the values of K^+m or K^-m are in the critical regions, then the hypothesis that the sequence is random must be rejected.

A.2.4 Runs Up and Down Test.

The last of the three tests selected was the runs up and down test. This test is examined in detail by both Knuth [25] and Fishman [28]. The associated test statistic is calculated based on the number of runs up and the number of runs down. Here, a run is defined as a series of observations such that $X_i < X_{i+1} < \ldots < X_{i+r}$ for runs up, or conversely, $X_j > X_{j+1} > \ldots > X_{j+s}$ for runs down, for $r,s>0$. The test statistic is given by

$$R = \sum_{i=1}^{P} \sum_{j=1}^{P} C_{ij} \quad [R_i - E(R_i)]\ [R_j - E(R_j)]$$

where R_i = number of runs of length i,
R_j = number of runs of length j,
$E(R_i)$ = expected number of runs of length i (see Table A.1),
$E(R_j)$ = expected number of runs of length j (see Table A.1),
C_{ij} = element in row i and column j of the inverse of the covariance matrix of R_1, ..., R_p (see Table A.2),
p = length of longest run.

TABLE A.1 (from Fishman [28])

$$E(R_i) = 2n\frac{i^2 + 3i + 1}{(i + 3)!} - 2\frac{i^3 + 3i^2 - i - 4}{(i + 3)!}$$

$=0.4167n$	$+ 0.0833$	$i=1$
$=0.1833n$	$- 0.2333$	$i=2$
$=0.0528n$	$- 0.1306$	$i=3$
$=0.0115n$	$- 0.0413$	$i=4$
$=0.0020n$	$- 0.0095$	$i=5$
$=0.0003n$	$- 0.0017$	$i=6$
$=3.9\times10^{-5}n$	$- 0.0003$	$i=7$

TABLE A.2

$$C = \begin{array}{cccccc} 4529.4 & 9044.9 & 13568 & 18091 & 22615 & 27892 \\ 9044.9 & 18097 & 27139 & 36187 & 45234 & 55789 \\ 13568 & 27139 & 40721 & 54281 & 67852 & 83685 \\ 18091 & 36187 & 54281 & 72414 & 90470 & 111580 \\ 22615 & 45234 & 67852 & 90470 & 113262 & 139476 \\ 27892 & 55789 & 83685 & 111580 & 139476 & 172860 \end{array}$$

The test statistic R is known to have an asymptotically chi-square distribution with p degrees of freedom [28]. Fishman proposes an analogous form using a six degree of freedom chi-square distribution for either of the cases where p = 5 or p = 7. This form combines R_7 with R_6 and $E(R_7)$ with $E(R_6)$. When p is equal to five, R_6 is set to zero so that the computer program used for the testing need not be altered.

A.3 TESTING

A.3.1 Judgment Criteria.

For the analysis of the "goodness" of a pseudo random number generator, the criteria given by Knuth [25] were used. The criteria specify that for the range of a given statistic S, a generator is classified as rejected if the value computed for a sample, S*, lies in the outermost two percent of the known distribution function of S (one percent on each end). Likewise, it is classified as "suspect" if S* lies in the next innermost eight percent and "almost suspect" if it lies in the next innermost ten percent. The following table summarizes these criteria.

TABLE A.3. ACCEPTANCE INDICATORS VERSUS TEST STATISTICS

S* in Range of S	Indication
0-1 percent, 99-100 percent	Reject
1-5 percent, 95-99 percent	Suspect
5-10 percent, 90-95 percent	Almost Suspect

Translating this table to the particular tests being used gives critical regions as shown in Table A.4.

One additional consideration should be examined concerning the use of multiple tests. For a rejection region of size alpha using N tests, the probability of rejecting a generator even though the hypothesis of randomness is true is given by $1 - (1\text{-alpha})^N$. Here alpha = 0.02 and N = 3, so the probability of rejecting a generator that is actually random enough is $1 - (1-0.02)^3 = 0.06$; therefore, the criteria of rejection used in this study lead to a 94 percent confidence level.

A.3.2 Test Procedures.

The ten UNIVAC GPSS 1100 pseudo random number generators were tested along with that of IBM GPSS/360. GPSS/360 actually has eight generators available, but when they are used in unmodified form, each returns an identical sequence of random numbers [1, p.144]. The UNIVAC generators were tested by using the GPSS 1100 random number generation algorithm to produce a sequence of numbers. Using the algorithm, instead of merely copying a sequence of numbers from a GPSS 1100 program, saved considerable time. It should be noted that the sequence generated by this approach was checked against the output of actual random numbers from a GPSS 1100 program to insure exact replication of the sequences. Unfortunately, this approach could not be easily applied to the IBM generator. This prompted the writing of a short program in GPSS/360 in

TABLE A.4. ACCEPTANCE INDICATORS VERSUS TEST STATISTIC CRITICAL REGIONS

1. Runs above and below the median	
$\|Z\| \geq 2.33$	Reject
$2.33 > \|Z\| \geq 1.65$	Suspect
$1.65 > \|Z\| \geq 1.28$	Almost Suspect
2. Maximum of five	
$K600 \leq 0.0648$	Reject
$K600 \geq 1.5092$	
$0.648 < K600 \leq .1544$	Suspect
$1.5092 > K600 \geq 1.2170$	
$K200 \leq .0603$	Reject
$K200 \geq 1.5033$	
$0.0603 < K200 \leq .1502$	Suspect
$1.5033 > K200 \geq 1.2119$	
3. Runs up and down	
$R \leq .872$	Reject
$R \geq 16.81$	
$.872 < R \leq 1.64$	Suspect
$16.81 > R \geq 12.59$	
$1.64 < R < 2.20$	Almost Suspect
$12.59 > R \geq 10.65$	

order to provide a listing of the IBM sequence, which was then read into the testing program. Only the results of tests for sequences of length 1000 to 3000 are discussed in detail because the simulation models of concern in this study call on any given generator approximately that many times in any run. Tests on sequences of length greater than 5000 are of little interest at this point, but some results of tests on these longer sequences are given in Table A.6. A discussion of the random number generation techniques is given in the next section.

A.4 GPSS PSEUDO RANDOM NUMBER GENERATION SCHEMES

A.4.1 IBM.

According to the IBM GPSS/360 User's Manual [2, pp. 36-37], the random number generation algorithm is as follows:

1. The appropriate word of the index array points to one of the eight numbers in the base number array. Since the index array words are initially zero, the first base number used will be the seed.

2. The appropriate number in the multiplier array is multiplied by the base number chosen in step 1.

3. The low-order 31 bits of this product are stored in the appropriate word of the multiplier array, to be used the next time a random number is called for.

4. Three bits of the high-order 16 bits of the product produced in step 2 are stored in the appropriate word of the index array, for future use. This number (0-7) points to one of eight words of the base number array to be used the next time a random number is called for.

5. (a) If the random number required is a fraction, the middle 32 bits of the product produced in step 2 are divided by 10^6, and the remainder becomes the six-digit fractional random number.

(b) If the random number required is an integer, the middle 32 bits of the product produced in step 2 are divided by 10^3, and the remainder becomes the three-digit random number.

A.4.2 UNIVAC.

The UNIVAC random number generation algorithm [3, pp.3.30, 3.32] is a simple one. It uses a linear congruential or mixed linear congruential generator, as the case may be. It takes the form

$$X_1 = S$$
$$X_{n+1} = (mX_n + I) \bmod 2^{35}$$

where S= seed, m= multiplier, and I= increment. When a fractional number is needed, the integer X_i is divided by 2^{35}. When an integer value from 0 to 999 is required, the fractional number is multiplied by 10^3 and truncated.

A.4.3 Independent Streams of Random Numbers.

The UNIVAC pseudo random number generator uses ten different combinations of multipliers, increments, and seeds to produce its ten random number sequences. The IBM has one generator, replicated eight times.

A.5 RESULTS OF TESTS

Using the established critical regions, it can be seen that most of the generators fared well. (See Table A.5.) It appears that UNIVAC generator nine may have a few problems associated with its use; the values of the runs up and down test statistics for sequence sizes of both 1000 and 3000 lie in the rejection region. Also, the value of the maximum of five test statistic K^-600 places more suspicion on the sequence produced by this generator. These facts suggest that generator nine should not be used, at least in short simulation models, because the number sequence produced by it does not exhibit sufficient randomness.

The only other generators with test statistic values in the rejection region are the UNIVAC generators one and two. The runs up and down test statistic for a sequence length of 1000 is far too large for each of the generators. It is interesting to note that generator one is used as the resident generator in the GPSS 1100 language. This means that on occasions when the TIME and GO TO fields require a random number, they call on generator one. (It should also be noted that the simulation models studied did not include these types of TIME and GO TO fields.) Generator two, which should also be rejected for a sequence size of 1000 according to Knuth's criteria, was employed in all four of the UNIVAC simulation models studied. For each message introduced into the model, the generator was called on twice, once to generate Poisson interarrivals, and once to create exponentially distributed message lengths. Since a minimum of 1000 messages were included in each run, the second generator was called on at least 2000 times, probably closer to 3000 times when "warmup" and queued messages are counted. Therefore, the nonrandom behavior of generator two for a sequence size of 1000 does not appear to be a possible cause for the discrepancy between the UNIVAC and IBM results.

The only other generator that is reasonably suspicious is the third UNIVAC generator. Three of the four maximum of five test statistics for sequences from this generator lie in the "suspicion" range. Incidentally, this is the generator used in the uniform distribution function in the UNIVAC models used to determine the routing of the messages.

Since only two random number generators are required for the UNIVAC simulation models in addition to generator one, it would seem advantageous to select generators that cast the least doubt on the results. This usage of the "best" generators would lead to a more meaningful comparison between UNIVAC and IBM data.

There appears to be no need to tamper with the IBM generator as it comes through the tests very well. But, if longer sequences are accepted for UNIVAC, then IBM sequences of similar length should be tested for randomness.

Examination of even longer sequences for the UNIVAC generators (see Table A.6) shows a trend for almost all of the generators failing the runs above and below the median test for sequence sizes greater than 10,000 numbers. The maximum of five test and the runs up and down test reject generators seven and six, respectively, for sequences of 8000 numbers and up. From these results, it can be seen that there are particular generators that should be avoided for certain sequence sizes.

A.6 SUMMARY AND CONCLUSION

The randomness tests performed indicate that the UNIVAC generators are primarily suited for models requiring numerical sequences of length from 3000 to somewhere around 8000. The IBM generator cannot be rejected at the 94 percent confidence level for sequences of length 1000 or 3000, but a study of its characteristics for longer sequences should be performed. From this study, it appears the generators used in the simulation models are in fact random enough and do not cause the principal differences between UNIVAC and IBM simulation results.

TABLE A.5 SUMMARY OF RANDOM NUMBER TESTS

GENERATOR	SEQUENCE SIZE	MEAN	MEDIAN	Z_N	$K^+(N/5)$	$K^-(N/5)$	R(N)
UNIVAC 1	1000	505	513	0.13	0.7739	0.3101	31.74***
	3000	502	509	-0.22	0.8147	0.4625	7.22
UNIVAC 2	1000	484	475	0.63	0.7957	0.1212**	33.15***
	3000	496	492	0.07	0.9345	0.1802	10.52
UNIVAC 3	1000	497	499	-1.27	0.9341	0.0891**	2.94
	3000	495	492	-1.02	1.3372**	0.0727**	7.36
UNIVAC 4	1000	498	499	-0.25	0.8428	0.1650	15.81**
	3000	491	487	0.04	0.9271	0.6819	1.17**
UNIVAC 5	1000	499	488	-1.71**	0.8114	0.2349	4.41
	3000	510	511	-1.94**	0.3244	1.1122	5.24
UNIVAC 6	1000	506	509	1.39*	0.2052	0.8640	6.68
	3000	492	491	1.50*	0.7937	0.2979	2.82
UNIVAC 7	1000	484	469	0.76	1.1430	0.3604	5.00
	3000	499	496	1.17	0.8742	0.6500	5.48
UNIVAC 8	1000	499	514	-0.70	0.8881	0.5400	12.71**
	3000	496	494	0.07	0.5209	1.0026	2.99
UNIVAC 9	1000	516	517	0.00	0.5809	0.9238	34.12***
	3000	501	508	0.84	0.3287	1.3742**	19.69***
UNIVAC 10	1000	502	505	-1.90**	0.5561	1.0451	4.33
	3000	495	498	-1.20	0.9188	0.6737	2.19*
IBM	1000	497	484	-1.45*	0.9675	0.3310	11.07*
	3000	493	490	0.44	1.0176	0.1103**	7.55

* Almost suspect
** Suspect
*** Reject

TABLE A.6 SUMMARY OF RANDOM NUMBER TESTS

GENERATOR NUMBER	NUMBER OF GENERATED R.N.'S	BASIC SERIES STATISTICS		RUNS ABOVE AND BELOW THE MEDIAN TEST	MAXIMUM OF 5 TEST		RUNS UP AND DOWN TEST
	N	MEAN	MEDIAN	Z_N	$K^+(N/5)$	$K^-(N/5)$	R(N)
1	3,000	502	509	-0.22	0.8147	0.4625	7.22
	8,000	502	508	-1.28	0.4339	1.2043	6.16
	10,000	502	507	-2.99	0.4013	1.1295	7.88
	12,000	501	506	1.71	0.2967	1.2199	11.70
2	3,000	496	492	0.07	0.9345	0.1802	10.52
	8,000	495	494	-0.64	0.7178	0.4526	9.88
	10,000	495	494	0.51	0.8555	0.6793	10.63
	12,000	496	494	4.02	0.7478	0.6870	5.77
3	3,000	495	492	1.02	1.3372	0.0727	7.36
	8,000	498	500	-1.76	0.6951	0.1955	10.84
	10,000	497	499	-0.63	0.7425	0.1473	7.81
	12,000	498	501	-5.72	0.6336	0.5711	9.87
4	3,000	491	487	0.04	0.9271	0.6819	1.17
	8,000	493	492	0.54	0.9859	0.2277	2.08
	10,000	494	493	-0.89	0.8466	0.1319	1.74
	12,000	492	491	-6.70	0.9625	0.1170	9.37
5	3,000	510	511	-1.94	0.3244	1.1122	5.24
	8,000	498	496	-5.77	0.8454	0.1703	4.96
	10,000	500	497	-4.45	0.7345	0.3696	2.39
	12,000	498	496	*****	0.7638	0.4029	4.39
6	3,000	492	491	1.50	0.7937	0.2979	2.82
	8,000	500	500	1.87	0.3055	0.8177	31.09
	10,000	499	497	3.75	0.4478	0.6726	32.09
	12,000	497	495	8.77	0.6531	0.5257	27.04
7	3,000	499	496	1.17	0.8742	0.6500	5.48
	8,000	505	506	2.14	0.1996	1.7293	6.10
	10,000	505	507	1.08	0.2232	1.8202	5.44
	12,000	506	508	3.53	0.2445	1.9327	12.76
8	3,000	496	494	0.07	0.5209	1.0026	2.99
	8,000	498	498	-3.90	0.9006	0.2238	5.76
	10,000	499	500	-6.48	0.9342	0.1887	8.86
	12,000	499	500	*****	1.0519	0.1110	7.33
9	3,000	501	508	0.84	0.3287	1.3742	19.69
	8,000	501	498	0.27	0.2688	1.4032	4.47
	10,000	501	499	-1.02	0.3801	1.2502	3.80
	12,000	500	498	2.92	0.3797	0.9532	12.78
10	3,000	495	498	-1.20	0.9188	0.6737	2.19
	8,000	500	504	-1.34	0.6798	0.4967	14.67
	10,000	499	503	-0.82	0.7262	0.2317	7.35
	12,000	500	505	1.10	0.7302	0.4668	10.09
				NORMAL	KOLMOGOROV-SMIRNOV		$X^2(6)$
IBM	1,000	497	484	-1.45	0.9675	0.3310	11.07
	3,000	493	490	0.44	1.0176	0.1103	7.55
	10,000	501	499	-0.32	0.6466	0.5921	1.32

Next page is blank.

APPENDIX B

GPSS/360 PROGRAM LISTINGS FOR RING NETWORK SIMULATIONS

Next page is blank.

For NEWHALL/IBM GPSS Program Listing see Reames [6], pp. 191-194.

The following blocks were inserted at the top of the program shown in Reames [6] in order to successfully execute the GPSS/360 program on the APG IBM 360/65 computer system:

```
REALLOCATE XAC,1200,BLO,100,FAC,100,STO,100,QUE,100,LOG,100
REALLOCATE TAB,50,FUN,10,VAR,20,FSV,100,HSV,50,CHA,100
REALLOCATE BVR,10,FMS,10,HMS,10,MAC,5,COM,90000
SIMULATE

             .
             .
             .
```

For the PIERCE/IBM program listing see Reames [6], pp. 187-190.

The change to this program starts at the bottom of page 189 in [6] and is as follows:

```
                        .
                        .
                        .

*  LAST PACKET OF A MESSAGE HAS BEEN RECEIVED. RECORD TOTAL
*  MESSAGE TRANSMISSION TIME.
*
 LASTP TABULATE   TMGTM            RECORD TOTAL MESSAGE TRANSIT TIME
*
*  CHECK IF LAST TERMINATION THEN SAVE RELATIVE CLOCK
*
       SAVEVALUE  3+,K1
       TEST E     X3,X4,PATW
       SAVEVALUE  2+,C1
*
*  SAVES VALUE OF RELATIVE CLOCK FOR ABSOLUTE CLOCK
*
 PATW  TERMINATE  1
*
*
*  TABLES AND QTABLES --
                        .
                        .
                        .
```

For DLCNNE/IBM Program Listing see Reames [6], pp. 178-186.

DLCNNE is identical to DLCN except that the following blocks in DLCN at the top of page 181 in [6], which now read:

```
                               .
                               .
                               .
 RECVR  LOGIC S     *1                GET CONTROL OF RECEIVER
        TRANSFER    .010,*+4,*+1      PERFORM MSG ERROR CHECKING,
        TRANSFER    .010,*+3,*+1      ASSUMING 1 ERROR PER 10,000 CHARS.
        ASSIGN      5,K3              IF ERROR, SET ACK MSG RESPONSE
        TRANSFER    ,RECVD            & GO SEND ACK MSG
        LOOP        6,RECVR+1         CHECK EACH CHAR. OF MSG FOR ERROR
*
 RECVD  ADVANCE     V$AMSG            ALLOW TIME TO RECEIVE DATA
                               .
                               .
                               .
```

have been changed in DLCNNE to read:

```
                               .
                               .
                               .
 RECVR  LOGIC S     *1                GET CONTROL OF RECEIVER
        TRANSFER    ,*+3              SKIP POSSIBILITY OF ERRORS IN CHARS.
        ASSIGN      5,K3              IF ERROR, SET ACK MSG RESPONSE
        TRANSFER    ,RECVD            & GO SEND ACK MSG.
        LOOP        6,RECVR+1         CHECK EACH CHAR. OF MSG FOR ERROR
*
 RECVD  ADVANCE     V$AMSG            ALLOW TIME TO RECEIVE DATA
                               .
                               .
                               .
```

This change disables retransmissions due to received character errors; hence, the name DLCN/"No Errors" or simply DLCNNE..

```
PL3/IBM
*
*
**
***    SIMULATION OF A 6 NODE RING PROCESSOR NETWORK
***    ALLOWING THE CONCURRENT GENERATION AND
***    TRANSMISSION OF ARBITRARY LENGTH MESSAGES THROUGH
***    THE USE OF A 'PLAYTHROUGH' PROTOCOL.
**
*
*
*  ASSUMPTIONS --
*
*  1)  MESSAGES ARE GENERATED IN EACH NODE BY INDEPENDENT POISSON
*      PROCESSES, EACH HAVING THE SAME MEAN ARRIVAL RATE.
*  2)  MESSAGE LENGTHS ARE ALSO EXPONENTIALLY DISTRIBUTED,
*      WITH A MEAN LENGTH OF 50 CHARACTERS AND A MAXIMUM
*      LENGTH OF 500 CHARACTERS.
*  3)  MESSAGES ALL HAVE AN ADDITIONAL 10 CHARACTERS OF
*      HEADER/TRAILER INFORMATION.
*  4)  MESSAGES ENCOUNTERING CONTENTION FOR THE RING ARE
*      RETRANSMITTED AT THE NEXT AVAILABLE OPPORTUNITY.
*
*  MESSAGE PARAMETER ASSIGNMENTS --
*
*  P1  DESTINATION NODE ADDRESS
*  P2  ORIGIN NODE ADDRESS
*  P3  CURRENT NODE ADDRESS
*  P4  TOTAL MESSAGE LENGTH (DATA + HEADER + TRAILER)
*  P5  MESSAGE DISTANCE
*
*
*  SAVEVALUES REPRESENTING NODE STATUS
*
*  X1 THROUGH X6
*
*      0 -- FREE
*      1 -- AWAITING ACKNOWLEDGE
*      2 -- SOURCE
*      3 -- BRIDGE
*
*
*  VARIABLE ASSIGNMENTS --
*
 NODE  VARIABLE   K6                     NUMBER OF NODES IN NETWORK
 DEST  BVARIABLE  P1'GE'P2
 DIFF1 VARIABLE   P1-P2
 DIFF2 VARIABLE   K6-P2+P1
 CDEST VARIABLE   P1@V$NODE+K1
 INCR  VARIABLE   (P3@V$NODE)+K1         INCREMENT CURRENT NODE
 MLEN  VARIABLE   K50                    MESSAGE LENGTH MULTIPLIER
 MIA   VARIABLE   K300                   MESSAGE IA RATE
 CHM1  VARIABLE   (CH*3)-1
*
*  FUNCTION DEFINITIONS --
*
 EXPON FUNCTION   RN2,C60                EXPONENTIAL IA TIME
.0,.0/.05,.05129/.10,.10536/.15,.16252/.20,.22314
```

```
.25,.28768/.30,.35667/.35,.43078/.40,.51083/.45,.59784
.50,.69315/.55,.79851/.575,.85567/.60,.91629/.625,.98083
.65,1.04982/.675,1.12393/.70,1.20397/.725,1.29098/.75,1.38629
.775,1.49165/.80,1.60944/.82,1.71480/.84,1.83258/.86,1.96611
.88,2.12026/.90,2.30259/.91,2.40795/.92,2.52573/.93,2.65926
.935,2.73337/.94,2.81341/.945,2.90042/.95,2.99573/.955,3.10109
.96,3.21888/.965,3.35241/.97,3.50656/.974,3.64966/.977,3.77226
.98,3.91202/.982,4.01738/.984,4.13517/.986,4.26870/.988,4.42285
.99,4.60517/.991,4.71053/.992,4.82831/.993,4.96185/.994,5.11600
.995,5.29832/.996,5.52146/.997,5.80914/.998,6.21461/.999,6.90776
.9995,7.601/.9998,8.52/.9999,9.21/.99995,9.9/1.0,10.0
*
 UNIF  FUNCTION   RN3,C2                UNIFORM DIST. OVER (1,5)
0,1/1,6
*
*
*  GENERATE THE 'GO' MESSAGE
*
       GENERATE    ,,,1,5,4,F           CREATE 1 COPY OF 'GO'
       ASSIGN      3,FN$UNIF            ASSIGN ARBITRARY STARTING POINT
 GOX   MARK        1                    MARK CONTROL PASSING TIME
       TEST E      X*3,K0,GOX1          SEE IF NODE IS FREE
       TEST NE     CH*3,K0,GOX1         SEE IF MESSAGES ON CHAIN
       LOGIC R     7                    RESET 'WAIT' LOGIC SWITCH
       SAVEVALUE   7,V$CHM1            INITIALIZE CHAIN COUNTER
       UNLINK      *3,SET3,1            REMOVE 1 MSG FROM CHAIN
       GATE LS     7                    STOP 'GO' UNTIL 'WAIT' SET
 GOX1  LOGIC S     *3                   SET TRANSMITTER'S LOGIC SWITCH
       PREEMPT     *3,PR                SEIZE  CURRENT NODE'S TRANSMITTER
       ADVANCE     1                    HOLD FOR TIME 1
       RETURN      *3                   RETURN CURRENT NODE'S TRANSMITTER
       LOGIC R     *3                   RESET TRANSMITTER'S LOGIC SWITCH
       TABULATE    CNLTM                TABULATE CONTROL PASSING TIME
       ASSIGN      3,V$INCR             INCREMENT NODE COUNTER
       TRANSFER    ,GOX                 ADVANCE TO NEXT NODE
*
*  INITIATE MESSAGES FROM EACH NODE EXPONENTIALLY
*
 MSG1  GENERATE    V$MIA,FN$EXPON,,,10,5,F      CREATE MSG AT NODE 1
       ASSIGN      2,K1                 SET MSG ORIGIN ADDRESS
       TRANSFER    ,SETUP               GO SET UP OTHER MSG PARAMETERS
*
 MSG2  GENERATE    V$MIA,FN$EXPON,,,10,5,F      CREATE MSG AT NODE 2
       ASSIGN      2,K2                 SET MSG ORIGIN ADDRESS
       TRANSFER    ,SETUP               GO SET UP OTHER MSG PARAMETERS
*
 MSG3  GENERATE    V$MIA,FN$EXPON,,,10,5,F      CREATE MSG AT NODE 3
       ASSIGN      2,K3                 SET MSG ORIGIN ADDRESS
       TRANSFER    ,SETUP               GO SET UP OTHER MSG PARAMETERS
*
 MSG4  GENERATE    V$MIA,FN$EXPON,,,10,5,F      CREATE MSG AT NODE 4
       ASSIGN      2,K4                 SET MSG ORIGIN ADDRESS
       TRANSFER    ,SETUP               GO SET UP OTHER MSG PARAMETERS
*
 MSG5  GENERATE    V$MIA,FN$EXPON,,,10,5,F      CREATE MSG AT NODE 5
       ASSIGN      2,K5                 SET MSG ORIGIN ADDRESS
       TRANSFER    ,SETUP               GO SET UP OTHER MSG PARAMETERS
*
```

```
MSG6  GENERATE   V$MIA,FN$EXPON,,,10,5,F      CREATE MSG AT NODE 6
      ASSIGN     2,K6                      SET MSG ORIGIN ADDRESS
*
*  SET OTHER MESSAGE PARAMETERS
*
SETUP ASSIGN     1,FN$UNIF                 SET MSG DEST. ADDRESS
      ASSIGN     1+,BV$DEST
SETZ  TEST G     P1,P2,SETA
      ASSIGN     5,V$DIFF1
      TRANSFER   ,XXXX
SETA  ASSIGN     5,V$DIFF2
XXXX  MSAVEVALUE TRAM+,*2,*1,K1,H          RECORD TRNS/RCVR ADDRESSING
      ASSIGN     4,V$MLEN,EXPON            SET MSG DATA LENGTH
      ASSIGN     4+,K4                     ADD HEADER/TRAILER FOR TOTAL
      TABULATE   MSGAR                     TABULATE MSG IA TIME
      TABULATE   MSGLN                     TABULATE MSG LENGTH
      ASSIGN     3,P2                      SET CURRENT NODE AS ORIGIN
      QUEUE      *3                        ENTER TRANSMISSION WAITING QUEUE
      LINK       *3,FIFO                   LINK MSG ON ORDER WAITING CHAIN
*
*  CHECK THAT THE DEST. IS WITHIN THE RANGE
*  OF THE ORIGIN.
*
SET3  TEST LE    MH$SRD(2,P3),P2,SET4
      TEST L     P1,P2,SETOK
      TRANSFER   ,SET5
SET4  TEST L     P1,P2,SET5
      TRANSFER   ,SETNG
SET5  TEST GE    MH$SRD(2,P3),P1,SETNG
      TRANSFER   ,SETOK
*
*  ANY TRANSACTION REACHING THE FOLLOWING SECTION
*  CANNOT BE SENT IMMEDIATELY, DUE TO INSUFFICIENT
*  RANGE OF ITS TRANSMITTER.  IT IS LINKED BACK
*  ONTO THE CHAIN TO BE RETRIED LATER.
*
SETNG TEST E     X7,K0,SETN1
      LOGIC S    7
      LINK       *3,FIFO
SETN1 SAVEVALUE  7-,K1
      UNLINK     *3,SET3,1
      LINK       *3,FIFO
*
*  TRANSACTIONS REACHING THE FOLLOWING SECTION ARE
*  WITHIN THE RANGE OF THEIR ORIGINS, AND ARE SENT.
*
SETOK DEPART     *3                        LEAVE TRANSMISSION WAITING QUEUE
      TEST E     P5,K1,TEST2
      TRANSFER   ,QQQ1
TEST2 TEST E     P5,K2,TEST3
      TRANSFER   ,QQQ2
TEST3 TEST E     P5,K3,TEST4
      TRANSFER   ,QQQ3
TEST4 TEST E     P5,K4,QQQ5
      TRANSFER   ,QQQ4
QQQ1  TABULATE   TLQ1
      TRANSFER   ,SETB
QQQ2  TABULATE   TLQ2
```

```
      TRANSFER   ,SETB
QQQ3  TABULATE   TLQ3
      TRANSFER   ,SETB
QQQ4  TABULATE   TLQ4
      TRANSFER   ,SETB
QQQ5  TABULATE   TLQ5
SETB  TABULATE   TLQTM                     TABULATE TOTAL QUEUEING TIME
      TABULATE   TRNAR                     TABULATE IA RATE OF SUCCESSES
      TEST E     X7,CH*3,SETO1
      SAVEVALUE  7,K0
      TRANSFER   ,SETO3
SETO1 TEST NE    X7,K0,SETO3
      UNLINK     *3,SETO2,1
      SAVEVALUE  7-,K1
      TRANSFER   ,SETO1
SETO2 LINK       *3,FIFO
SETO3 SAVEVALUE  *2,K2                     DECLARE NODE AS SOURCE
      PREEMPT    *2,PR                     SEIZE ORIGIN'S TRANSMITTER
      ADVANCE    3
      RETURN     *2                        RETURN ORIGIN'S TRANSMITTER
      SPLIT      1,BEGIN                   SEND COPY TO BEGIN
      PRIORITY   1                         LOWER PRIORITY
      LOGIC S    7                         SET 'WAIT' LOGIC SWITCH
      PREEMPT    *2,PR                     SEIZE ORIGIN'S TRANSMITTER
      ADVANCE    *4                        HOLD FOR DATA MSG TRANSMISSION
      RETURN     *2                        RETURN ORIGIN'S TRANSMITTER
      PRIORITY   10                        RAISE PRIORITY
      GATE LS    *2                        WAIT FOR 'GO'
      PREEMPT    *2,PR                     SEIZE ORIGIN'S TRANSMITTER
      ADVANCE    3                         ALLOW TIME TO SEND TRAILER
      RETURN     *2                        RETURN ORIGIN'S TRANSMITTER
      SAVEVALUE  *2,K1                     DECLARE NODE AS AWAITING ACK.
      TRANSFER   ,END1
*
*  SEE IF THERE ARE ANY OTHER MESSAGES ON THE
*  LOOP.  IF NOT, GO TO STRT.  IF SO, GO TO HEAD.
*
BEGIN TEST E     MH$SRD(2,P2),P2,HEAD
      TRANSFER   ,STRT
*
*  SEND THE MESSAGE, ALTERING SRC, RANGE, AND DEST.
*  REGISTERS ALONG THE WAY.  SET ALL INTERMEDIATE
*  NODES TO THE BRIDGE STATE.
*
STRT  MSAVEVALUE SRD,K1,*3,K0,H            SET SRC REGISTER
      MSAVEVALUE SRD,K3,*3,P1,H             SET DEST REGISTER
STRT1 ASSIGN     3,V$INCR                  INCREMENT NODE COUNTER
      TEST NE    P3,P1,SET                 IF NOT DEST YET,
      MSAVEVALUE SRD,K1,*3,*2,H            SET SRC REGISTER
      MSAVEVALUE SRD,K2,*3,*2,H            SET RANGE REGISTER
      MSAVEVALUE SRD,K3,*3,*1,H            SET DEST REGISTER
      SAVEVALUE  *3,K3                     DECLARE NODE AS BRIDGE
      PREEMPT    *3,PR                     SEIZE CURRENT NODE'S TRANS.
      ADVANCE    3                         ALLOW TIME TO SEND
      RETURN     *3                        RETURN TRANS.
      SPLIT      1,STRT2                   LET DATA MSG BE SENT
      TRANSFER   ,STRT1                    CONTINUE
*
```

```
*  HOLD EACH BRIDGE FOR THE LENGTH OF THE MSG
*
 STRT2 PRIORITY    1                         LOWER PRIORITY
       PREEMPT     *3,PR                     SEIZE CURRENT NODE'S TRANS.
       ADVANCE     *4                        ALLOW TIME TO SEND
       RETURN      *3                        RETURN TRANS.
       TERMINATE                             DESTROY COPY OF MSG
*
*  AT THE DEST., START IS TRANSFORMED INTO SET
*
 SET   MSAVEVALUE SRD,K1,*3,P2,H             SET SRC REGISTER
       MSAVEVALUE SRD,K2,*3,P2,H             SET RANGE REGISTER
       MSAVEVALUE SRD,K3,*3,K0,H             SET DEST REGISTER
*
*  SET RESETS RANGE REGISTERS UNTIL REACHING ORIGIN
*
 SET1  PREEMPT     *3,PR                     SEIZE CURRENT NODE'S TRANS.
       ADVANCE     3                         ALLOW TIME TO SEND
       RETURN      *3                        RETURN TRANS.
       ASSIGN      3,V$INCR                  INCREMENT NODE COUNTER
       TEST NE     P3,P2,SETFN
       MSAVEVALUE SRD,K2,*3,P2,H             SET RANGE REGISTER
       TRANSFER    ,SET1
*
*  AT SETFN,  SET HAS COMPLETED ITS TASK, AND IS DESTROYED
*
 SETFN TERMINATE
*
*  AT END1, THE DETERMINATION IS MADE AS TO WHETHER
*  TO USE STOP OR TAIL.
*
 END1  TEST E      MH$SRD(2,P2),P2,TAIL
       TRANSFER    ,STP
*
*  STOP MUST UNDO THE ACTION OF START
*
 STP   ASSIGN      3,V$INCR                  INCREMENT NODE COUNTER
       MSAVEVALUE SRD,K1,*3,K0,H             SET SRC REGISTER
       MSAVEVALUE SRD,K2,*3,P3,H             SET RANGE REGISTER
       MSAVEVALUE SRD,K3,*3,K0,H             SET DEST REGISTER
       TEST NE     P3,P1,RSET                IF NOT DEST YET,
       SAVEVALUE   *3,K0                     DECLARE NODE AS FREE
       PREEMPT     *3,PR                     SEIZE CURRENT NODE'S TRANS.
       ADVANCE     3                         ALLOW TIME TO SEND
       RETURN      *3                        RETURN TRANS.
       TRANSFER    ,STP                      CONTINUE
*
*  AT THE DEST., STOP BECOMES RESET
*
 RSET  PREEMPT     *3,PR                     SEIZE CURRENT NODE'S TRANS.
       ADVANCE     3                         ALLOW TIME TO SEND
       RETURN      *3                        RETURN TRANS.
       ASSIGN      3,V$INCR                  INCREMENT NODE COUNTER
       TEST NE     P3,P2,RESFN               IF NOT AT ORIGIN
       MSAVEVALUE SRD,K2,*3,P3,H             SET RANGE REGISTER
       TRANSFER    ,RSET                     CONTINUE
*
*
```

```
*  AT THIS POINT, RESET HAS REACHED THE ORIGIN.  IT WILL
*  RESET IT, AND REMOVE A TRANSACTION FROM THE SYSTEM.
*
 RESFN MSAVEVALUE SRD,K1,*3,K0,H       SET SRC REGISTER
       MSAVEVALUE SRD,K2,*3,P3,H       SET RANGE REGISTER
       MSAVEVALUE SRD,K3,*3,K0,H       SET DEST REGISTER
       SAVEVALUE  *3,K0                DECLARE NODE AS FREE
       TABULATE   TTLTM                TABULATE TOTAL TRANS. TIME
       TERMINATE  1                    DESTROY THIS MSG
*
*  'HEAD' CONTROL MSGS ENTER AT THIS POINT
*
 HEAD  MSAVEVALUE SRD,K3,*3,P1,H       SET DEST REGISTER
 HEAD1 ASSIGN     3,V$INCR             INCREMENT NODE COUNTER
       TEST NE    P3,P1,INIT           IF NOT DEST YET,
       MSAVEVALUE SRD,K1,*3,P2,H       SET SRC REGISTER
       MSAVEVALUE SRD,K3,*3,P1,H       SET DEST REGISTER
       SAVEVALUE  *3,K3                DECLARE NODE AS BRIDGE
       PREEMPT    *3,PR                SEIZE CURRENT NODE'S TRANS.
       ADVANCE    3                    ALLOW TIME TO SEND
       RETURN     *3                   RETURN TRANS.
       SPLIT      1,HEAD2              LET DATA MSG BE SENT
       TRANSFER   ,HEAD1               CONTINUE
*
*  HOLD EACH NODE FOR THE LENGTH OF THE MSG
*
 HEAD2 PRIORITY   1                    LOWER PRIORITY
       PREEMPT    *3,PR                SEIZE CURRENT NODE'S TRANS.
       ADVANCE    *4                   ALLOW TIME TO SEND
       RETURN     *3                   RETURN TRANS.
       TERMINATE                       DESTROY THE MESSAGE
*
*  AT INIT, THE AFFECTED REGION IS SEARCHED OUT
*
 INIT  MSAVEVALUE SRD,K1,*3,P2,H       SET SRC REGISTER
 INIT1 TEST E     X*3,K2,INIT2         IS THIS A SOURC
       TEST NE    MH$SRD(2,P3),MH$SRD(2,P2),INTX
 INIT2 PREEMPT    *3,PR                SEIZE CURRENT NODE'S TRANS.
       ADVANCE    3                    ALLOW TIME TO SEND
       RETURN     *3                   RETURN TRANS.
       ASSIGN     3,V$INCR             INCREMENT NODE COUNTER
       TRANSFER   ,INIT1               CONTINUE
*
*  NOW CHANGE INIT TO INIT*
*
 INTX  MSAVEVALUE SRD,K2,*3,P2,H       SET RANGE REGISTER
       PREEMPT    *3,PR                SEIZE CURRENT NODE'S TRANS.
       ADVANCE    3                    ALLOW TIME TO SEND
       RETURN     *3                   RETURN TRANS.
       ASSIGN     3,V$INCR             INCREMENT NODE COUNTER
       TEST NE    P3,P2,INTX1          IF AT NEW ORIGIN, THEN DONE
       TRANSFER   ,INTX                OTHERWISE CONTINUE
*
*  INIT* HAS COMPLETED ITS TASK
*
 INTX1 TERMINATE                       DESTROY MESSAGE
*
*  THE WORK OF TAIL IS NOW TO BE STARTED.  (MSG ENDING
*  ON A MULTIPLE MESSAGE LOOP)
*
 TAIL  MSAVEVALUE SRD,K3,*3,K0,H       SET DEST REGISTER
 TAIL1 ASSIGN     3,V$INCR             INCREMENT NODE COUNTER
       TEST NE    P3,P1,TERM           IF NOT DEST YET
       MSAVEVALUE SRD,K1,*3,K0,H       SET SRC REGISTER
       MSAVEVALUE SRD,K3,*3,K0,H       SET DEST REGISTER
       SAVEVALUE  *3,K0                DECLARE NODE AS FREE
       PREEMPT    *3,PR                SEIZE CURRENT NODE'S TRANS.
       ADVANCE    3                    ALLOW TIME TO SEND
       RETURN     *3                   RETURN TRANS.
       TRANSFER   ,TAIL1               CONTINUE
*
*  NOW AT DESTINATION.  CONVERT TAIL TO TERM
*  AND SEARCH FOR THE AFFECTED REGION.
*
 TERM  MSAVEVALUE SRD,K1,*3,K0,H         SET SRC REGISTER
 TERM1 TEST E     X*3,K2,TERM2         IS CURRENT NODE A SR
       TEST NE    MH$SRD(2,P3),P2,TRMX
*
*
*
 TERM2 PREEMPT    *3,PR                SEIZE CURRENT NODE'S TRANS.
       ADVANCE    3                    ALLOW TIME TO SEND
       RETURN     *3                   RETURN TRANS.
       ASSIGN     3,V$INCR             INCREMENT NODE COUNTER
       TRANSFER   ,TERM1
*
*  NOW CONVERT TERM TO TERM* AND RESET THE AFFECTED REGION
*
 TRMX  MSAVEVALUE SRD,K2,*3,MH$SRD(2,P2),H
       PREEMPT    *3,PR                SEIZE CURRENT NODE'S TRANS.
       ADVANCE    3                    ALLOW TIME TO SEND
       RETURN     *3                   RETURN TRANS.
       ASSIGN     3,V$INCR             INCREMENT NODE COUNTER
       TEST NE    P3,P2,TRMX1          IS THIS NEW ORIGI
       TRANSFER   ,TRMX                IF SO, CONTINUE
*
*  TERM* HAS COMPLETED ITS TASK
*
 TRMX1 SAVEVALUE  *3,K0                DECLARE NODE AS FREE
       TABULATE   TTLTM                TABULATE TOTAL TRANS. TIME
       TERMINATE  1                    DESTROY THIS MSG
*
*
*  DEFINE THE MATRICES, TABLES, ETC.
*
 TRAM  MATRIX     H,6,6                TRANS/RECVR ADDRESSING MATRIX
 SRD   MATRIX     H,3,6                REGISTER MATRIX
*
 TRNAR TABLE      IA,10,10,56          TOTAL XMITTED MSG IA RATE
 MSGLN TABLE      P4,10,10,56          TOTAL GENERATED MSG LENGTH
 MSGAR TABLE      IA,20,20,56          TOTAL GENERATED MSG IA RATE
 CNLTM TABLE      MP1,0,2,56           CONTROL PASSING TIME
 TLQTM TABLE      M1,0,200,56          TOTAL QUEUEING TIME
 TTLTM TABLE      M1,200,200,56        TOTAL MSG TRANSIT TIME
 TLQ1  TABLE      M1,0,100,200         QUEUE TIME FOR 1 NODE DX.
 TLQ2  TABLE      M1,0,100,200         2 NODES TRAVELLED
```

```
TLQ3  TABLE      M1,0,100,200         3 NODES TRAVELLED
TLQ4  TA3LE      M1,0,100,200         4 NODES TRAVELLED
TLQ5  TABLE      M1,0,100,200         5 NODES TRAVELLED
*
TQT1  QTABLE     1,0,200,56           WAITING TIME FOR TRANS 1
TQT2  QTABLE     2,0,200,56           WAITING TIME FOR TRANS 2
TQT3  QTABLE     3,0,200,56           WAITING TIME FOR TRANS 3
TQT4  QTABLE     4,0,200,56           WAITING TIME FOR TRANS 4
TQT5  QTABLE     5,0,200,56           WAITING TIME FOR TRANS 5
TQT6  QTABLE     6,0,200,56           WAITING TIME FOR TRANS 6
*
*
*  CONTROL CARDS FOR SIMULATION.
*
      SIMULATE
*
      INITIAL    MH$SRD(1,1-6),0
      INITIAL    MH$SRD(3,1-6),0
      INITIAL    MH$SRD(2,1),1/MH$SRD(2,2),2/MH$SRD(2,3),3
      INITIAL    MH$SRD(2,4),4/MH$SRD(2,5),5/MH$SRD(2,6),6
      INITIAL    X7,0/X1,0/X2,0/X3,0/X4,0/X5,0/X6,0
      INITIAL    LS7
      START      100
      RESET
      INITIAL    MH$TRAM(1-6,1-6),0
      START      1000
*
*  END OF SIMULATION.
*
      END
```

Next page is blank.

APPENDIX C

GPSS 1100 PROGRAM LISTINGS FOR RING NETWORK SIMULATIONS

Next page is blank.

```
NEWHALL/Q
    JOB
    ORDER,P     6
    ORDER,S     6
    ORDER,L     6
    ORDER,F     6
    ORDER,Q     6
*
**
***  SIMULATION OF A 6-NODE NEWHALL LOOP NETWORK WHICH USES A
***  CONTROL-PASSING MECHANISM TO TRANSMIT VARIABLE-LENGTH
***  MESSAGES WHOSE LENGTHS ARE EXPONENTIALLY DISTRIBUTED WITH
***  A MEAN LENGTH OF 50 CHARACTERS.  EACH MESSAGE ALSO
***  INCLUDES 9 CHARACTERS OF HEADER INFORMATION.
**
*
*
*  MESSAGE PARAMETER ASSIGNMENTS --
*
*  P1             DESTINATION NODE ADDRESS
*  P2             ORIGINATION NODE ADDRESS
*  P3             CURRENT NODE ADDRESS
*  P4             TOTAL MESSAGE LENGTH
*  P5             (UNUSED)
*  P6             TRANSIT TIMER
*
*
*  VARIABLE DEFINITIONS --
*
DEST BVARIABLE  P$P(1) GE P$P(2)
INCR VARIABLE   (P$P(3)//6)+1
REST VARIABLE     P$P(4)-1
M.I.A VARIABLE 600
M.LN  VARIABLE 50
V1   VARIABLE     P$P(1)
V2   VARIABLE     P$P(2)
V3   VARIABLE     P$P(3)
V4   VARIABLE     P$P(4)
V5   VARIABLE     P$P(5)
V6   VARIABLE     P$P(6)
*
*
*  FUNCTION DEFINITIONS --
*
EXPON FUNCTION,EXP RF$2,1,1
*
UNIF FUNCTION,UNI RF$3,1,5
*
*
*  STORAGES, TABLES & QTABLES.
*
S(1) CAPACITY    1
S(2) CAPACITY    1
S(3) CAPACITY    1
S(4) CAPACITY    1
S(5) CAPACITY    1
S(6) CAPACITY    1
*
```

```
CNLTM             TABLE MP$P(1),5.,15.,56
MSGAR             TABLE IA.20.,20.,56
MSGLN             TABLE P$P(4),10.,10.,53
RSWTM             TABLE M$1,0.,100.,56
TQWTM             TABLE MP$P(6),0.,50.,56
TLQTM             TABLE M$1,0.,200.,56
TRNTM             TABLE MP$P(6),10.,10.,56
TMGTM             TABLE M$1,200.,200.,56
*
TWQ1 QTABLE       0.,200.,56,Q(1)
TWQ2 QTABLE       0.,200.,56,Q(2)
TWQ3 QTABLE       0.,200.,56,Q(3)
TWQ4 QTABLE       0.,200.,56,Q(4)
TWQ5 QTABLE       0.,200.,56,Q(5)
TWQ6 QTABLE       0.,200.,56,Q(6)
*
*
*  GENERATE SINGLE MESSAGE TO IMPLEMENT CONTROL PASSING MECHANISM.
*
*
     GENERATE     0,1
CNTRL             ASSIGN P(3),V$INCR
     ADVANCE      TIME(1)
     MARK         P(1)      * SHOULD THIS PERHAPS BE P(6)
     SEIZE        F(V$V3)
*
     ADVANCE      GOTO(+1,+4)
      COMPARE      Q$Q(V$V3) NE 0
     LOGIC        S,L(V$V3)
     GATE         LR,L(V$V3)
*
     ADVANCE      TIME(1)
     RELEASE      F(V$V3)
     TABULATE     CNLTM
     ADVANCE      GOTO(CNTRL)
*
*
*  GENERATE 6 INDEPENDENT IDENTICAL MESSAGE SOURCES.
*
MSG1 GENERATE 0     TIME(V$M.I.A*FN$EXPON)
     ASSIGN       P(2),1
     ADVANCE      GOTO(SETUP)
*
MSG2 GENERATE 0      TIME(V$M.I.A*FN$EXPON)
     ASSIGN       P(2),2
     ADVANCE      GOTO(SETUP)
*
MSG3 GENERATE 0      TIME(V$M.I.A*FN$EXPON)
     ASSIGN       P(2),3
     ADVANCE      GOTO(SETUP)
*
MSG4 GENERATE 0     TIME(V$M.I.A*FN$EXPON)
     ASSIGN       P(2),4
     ADVANCE      GOTO(SETUP)
*
MSG5 GENERATE 0      TIME(V$M.I.A*FN$EXPON)
     ASSIGN       P(2),5
     ADVANCE      GOTO(SETUP)
```

```
*
MSG6 GENERATE 0      TIME(V$M.I.A*FN$EXPON)
     ASSIGN       P(2),6
*
*
*  SET DESTINATION & CURRENT ADDRESSES.  SET MESSAGE LENGTH.
*  RECORD INTERARRIVAL RATE AND MESSAGE LENGTH DISTRIBUTIONS.
*
SETUP             ASSIGN P(1),FN$UNIF
     ASSIGN       P(1),P$P(1)+BV$DEST
     ASSIGN       P(3),P$P(2)
     ASSIGN       P(4),V$M.LN*FN$EXPON
     ASSIGN       P(4),P$P(4)+9
     TABULATE     MSGAR
     TABULATE     MSGLN
*
*
*  ADD MESSAGE TO TRANSMISSION QUEUE & WAIT FOR CONTROL TO
*  BE PASSED TO THIS NODE.  THEN TRANSMIT ALL MESSAGES IN
*  THE QUEUE ONTO THE LOOP.  WHEN THE QUEUE IS EMPTY, ALLOW
*  CONTROL TO PASS ON TO THE NEXT NODE.
*
TRANS INQUEUE Q(V$V2),P(5)
     ENTER        S(V$V2)
*
     TABULATE     RSWTM
     MARK         P(6)
     GATE         LS,L(V$V2)
*
      OUTQUEUE Q(V$V2),P(5)
     TABULATE     TQWTM
     TABULATE     TLOTM
     MARK         P(6)
     ADVANCE      TIME(1)
     SPLIT        1,RECVR
*
     ADVANCE      TIME(V$REST)
     LOGIC        R,L(V$V2)
     LEAVE        S(V$V2)
     TERMINATE
*
*
*  RELAY MESSAGE FROM RECEIVER TO RECEIVER UNTIL PROPER DESTINATION
*  IS REACHED, THEN REMOVE MESSAGE FROM THE LOOP.
*
RECVR             ASSIGN P(3),V$INCR
     ADVANCE      TIME(1)
     ADVANCE      GOTO(+1,RCVD)
     COMPARE      P$P(1) NE P$P(3)
*
     SEIZE        F(V$V3)
     ADVANCE      TIME(1)
     SPLIT        1,RECVR
*
     ADVANCE      TIME(V$REST)
     RELEASE      F(V$V3)
     TERMINATE
*
```

```
*
*  MESSAGE IS AT ITS DESTINATION.  REMOVE IT & COLLECT STATISTICS.
*
RCVD ADVANCE     TIME(V$REST)
      TABULATE    TRNTM
      TABULATE    TMGTM
     TERMINATE,R 1
*
*
*
*  SIMULATION CONTROL CARDS.
*
*
      START,NP        200
      RESET
      START       1000
*
      CLEAR
M.I.A VARIABLE 420
      START,NP        200
      RESET
      START       1000
*
      CLEAR
M.I.A VARIABLE 300
      START,NP        200
      RESET
      START       1000
*
      CLEAR
M.I.A VARIABLE 500
      START,NP        200
      RESET
      START       1000
*
      END
```

```
PIERCE/Q
     JOB
     ORDER.P  8
     ORDER.F  6
     ORDER.Q  6
      ORDER.X CLOCK.TERM.COUNT
**
*** SIMULATION OF A 6-NODE PIERCE LOOP WHICH USES A SINGLE
*** PACKET (9CHARACTER HEADER. REST DATA) FOR TRANSMISSION OF
*** VARIABLE-LENGTH MESSAGES (LENGTH EXPONENTIALLY DISTRIBUTED
*** WITH A MEAN LENGTH OF 50 CHARACTERS).  THE PACKET SIZE
*** (INCLUDING HEADER) IS SET BY VARIABLE 'PKLN' TO 36 CHARS.
**
*
* MESSAGE PARAMETER ASSIGNMENTS:
*  P1  DESTINATION NODE ADDRESS
*  P2  ORIGINATION NODE ADDRESS
*  P3  CURRENT NODE ADDRESS
*  P4  REMAINING MESSAGE LENGTH
*  P5  PACKET DATA LENGTH
*  P6  PACKET SYNCHRONIZATION TIME
*  P7  TRANSIT TIMER
*  P8  PACKET TIMER
*
*   VARIABLE DEFINITIONS
*
DEST BVARIABLE P$P(1) GE P$P(2)  ● 0 IF DEST < ORIGIN. 1 OTHERWISE
PKLN VARIABLE  36
PSYN  VARIABLE  2*(P$P(2)-1)-V$ACLK//V$PKLN   ● PACKET SYNCHRONIZATION TIME
ACLK  VARIABLE C$1+X$CLOCK
INCR VARIABLE  (P$P(3)//6)+1
DTLN VARIABLE  V$PKLN-9
NPKT VARIABLE  (P$P(4)-1)/V$DTLN+1  ● NUMBER OF PACKETS THIS MESSAGE
WCHR VARIABLE  V$DTLN-P$P(5)
PKLN1 VARIABLE  V$PKLN-1
ABOX VARIABLE  V$PKLN-12
M.I.A VARIABLE 300
M.LN  VARIABLE 50
V1     VARIABLE     P$P(1)
V2     VARIABLE     P$P(2)
V3     VARIABLE     P$P(3)
V4     VARIABLE     P$P(4)
V5     VARIABLE     P$P(5)
V6     VARIABLE     P$P(6)
V7     VARIABLE     P$P(7)
V8     VARIABLE     P$P(8)
*
*  TABLES AND QTABLES ***
*
MSGAR TABLE  IA,15,,15,,56
MSGLN TABLE  P$P(4),10,,10,,53
NPKMG TABLE  V$NPKT,1,,1,,35
WCHPK TABLE  V$WCHR,0,,2,,50
*
SYNTM TABLE  M$1,0,,2,,50
PKWTM TABLE  MP$P(7),0,,72,,55
PTRTM TABLE  MP$P(7),9,,3,,55
TPKTM TABLE  MP$P(8),20,,20,,56
```

```
TMGTM TABLE  M$1,20.,20.,56
*
TWQ1 QTABLE  0.,72.,55,Q(1)
TWQ2 QTABLE  C.,72.,55,Q(2)
TWQ3 QTABLE  0.,72.,55,Q(3)
TWQ4 QTABLE  0.,72.,55,Q(4)
TWQ5 QTABLE  0.,72.,55,Q(5)
TWQ6 QTABLE  0.,72.,55,Q(6)
*
*  FUNCTION DEFINITIONS --
*
EXPON FUNCTION,EXP RF$2,1,1
*
UNIF FUNCTION,UNI RF$3,1,5
*
*
*  GENERATE MESSAGES FROM EACH OF 6 INDEPENDENT NODES.
*
      INITIAL TERM,250
MSG1  GENERATE 0  TIME(V$M.I.A*FN$EXPON)
      ASSIGN P(2),1
      ADVANCE GOTO(SETUP)
*
MSG2  GENERATE 0  TIME(V$M.I.A*FN$EXPON)
      ASSIGN P(2),2
      ADVANCE GOTO(SETUP)
*
MSG3  GENERATE 0  TIME(V$M.I.A*FN$EXPON)
      ASSIGN P(2),3
      ADVANCE GOTO(SETUP)
*
MSG4  GENERATE 0  TIME(V$M.I.A*FN$EXPON)
      ASSIGN P(2),4
      ADVANCE GOTO(SETUP)
*
MSG5  GENERATE 0  TIME(V$M.I.A*FN$EXPON)
      ASSIGN P(2),5
      ADVANCE GOTO(SETUP)
*
MSG6 GENERATE 0  TIME(V$M.I.A*FN$EXPON)
      ASSIGN P(2),6
      ADVANCE GOTO(SETUP)
*
*
*  SET DESTINATION ADDRESS & MESSAGE LENGTH.  CALCULATE TIME TO NEXT
*  PACKET INTERVAL & SYNCHRONIZE WITH START OF IT.
*
SETUP ASSIGN   P(1),FN$UNIF @RANDOMLY ASSIGN DEST. ADDRS.
     ASSIGN   P(1),P$P(1)+BV$DEST
     ASSIGN   P(3),P$P(2)
     ASSIGN   P(4),V$M.LN*FN$EXPON
     TABULATE MSGAR
     TABULATE MSGLN
     TABULATE NPKMG
*
     ASSIGN   P(6),V$PSYN
     ADVANCE GOTO(+1,TSYNC)
     COMPARE  P$P(6) NE 0
     ADVANCE GOTO(+1,+3)
     COMPARE  P$P(6) G 600
      ASSIGN P(6),131071-P$P(6)+1
     ASSIGN   P(6),P$P(6)+V$PKLN
     ADVANCE TIME(V$V6)
*
TSYNC TABULATE  SYNTM
     ADVANCE GOTO(GNPKT)
*
*
*   CREATE ONE PACKET AT EACH PACKET INTERVAL & QUEUE IT FOR TRANS-
*   MISSION UNTIL THE MESSAGE GENERATED HAS BEEN COMPLETED.
*
GENPK ADVANCE   TIME(V$PKLN)
*
GNPKT ASSIGN   P(5),P$P(4)
     ASSIGN   P(4),P$P(4)-V$DTLN
     ADVANCE  GOTO(+1,QPKTS)
     COMPARE P$P(4) G 0
      ADVANCE GOTO(+1,QPKTS)
      COMPARE P$P(4) L 600
     ASSIGN   P(5),V$DTLN
     SPLIT    1,GENPK
*
QPKTS INQUEUE   Q(V$V2),PQ
     TABULATE   WCHPK
     MARK   P(7)
     MARK   P(8)
*
*  WAIT FOR TRANSMITTER TO BE FREE AT START OF PACKET INTERVAL,
*  THEN SEIZE IT LONG ENOUGH TO TRANSMIT ONE PACKET ONTO THE LOOP.
*
WTPKT ADVANCE GOTO(+1,TRNPK)
     GATE U,F(V$V2)
     ADVANCE  TIME(V$PKLN)
     ADVANCE  GOTO(WTPKT)
*
TRNPK OUTQUEUE  Q(V$V2),PQ
      PRIORITY 1
     TABULATE   PKWTM
     MARK  P(7)
*
*  ALLOW FOR TRANSMITTER DELAY, THEN SEND MESSAGE ON TO NEXT NODE.
*  THEN RELEASE TRANSMITTER AT END OF PACKET INTERVAL.
*
RELAY SEIZE  F(V$V3)
     ADVANCE  TIME(1)
     SPLIT  1,BBOX
     ADVANCE  TIME(V$PKLN1)
     RELEASE  F(V$V3)
     TERMINATE
*
*  CHECK IF NEXT B-BOX IS MESSAGE DESTINATION.  IF SO, REMOVE MESSAGE.
*  IF LAST B-BOX, INSERT A-BOX DELAY FOR PACKET SYNCHRONIZATION.
*
BBOX ASSIGN  P(3),V$INCR
     ADVANCE  TIME(1)
     ADVANCE  GOTO(+1,+3)
```

```
      COMPARE  P$P(3) EQ 1
      ADVANCE  TIME(V$ABOX)
      ADVANCE  GOTO(+1,RELAY)
      COMPARE  P$P(3) EQ P$P(1)
*
*  A PACKET HAS ARRIVED AT ITS FINAL DESTINATION.  REMOVE IT
*  AND CALCULATE APPROPRIATE STATISTICS.
*
RCVR ADVANCE  TIME(8)
      ADVANCE  TIME(V$V5)
      TABULATE  PTRTM
      TABULATE  TPKTM
      ADVANCE  GOTO(+1,LASTP)
      COMPARE  P$P(4) G 0
      TERMINATE
*
*  LAST PACKET OF A MESSAGE HAS BEEN RECEIVED.  RECORD TOTAL
*  MESSAGE TRANSMISSION TIME.
*
LASTP TABULATE  TMGTM
       SAVEX COUNT,X$COUNT+1
       ADVANCE GOTO(+1,PATW)
       COMPARE X$COUNT EQ X$TERM
       SAVEX  CLOCK,X$CLOCK+C$1
PATW TERMINATE,R  1
*
*  CONTROL CARDS  **
*
       START 200,NP
       RESET
       START 1200
       END
```

```
DLCN/Q
          JOB
     ORDER,P     9
     ORDER,F     6
     ORDER,Q     6
     ORDER,L     6
     ORDER,S     6
*
**
***      SIMULATION OF A 6-NODE DISTRIBUTED LOOP COMPUTER NETWORK (DLCN)
***      WHICH ALLOWS CONCURRENT GENERATION & TRANSMISSION OF
***      VARIABLE-LENGTH MESSAGES.
**
*
*  ASSUMPTIONS --
*
*  1)  MESSAGES ARE GENERATED IN EACH NODE BY INDEPENDENT POISSON
*      PROCESSES, EACH HAVING THE SAME MEAN ARRIVAL RATE,
*  2)  MESSAGE LENGTHS ARE ALSO EXPONENTIALLY DISTRIBUTED, WITH A MEAN
*      LENGTH OF 50 CHARACTERS AND A MAXIMUM LENGTH OF 500 CHARACTERS.
*  3)  MESSAGES ALL HAVE AN ADDITIONAL 9 CHARACTERS OF HEADER/TRAILER
*      INFORMATION (DESTINATION ADDRESS, ORIGIN ADDRESS, LENGTH,
*      CONTROL, ERROR CHECKING, ETC.) APPENDED TO EACH MESSAGE.
*  4)  EACH DATA MESSAGE RECEIVED IS ACKNOWLEDGED TO ITS SENDER
*      (NO RESPONSE, RECEIVER BUSY, ACCEPTED, ERROR) BY TRANSMISSION
*      OF A 6-CHARACTER ACKNOWLEDGEMENT MESSAGE.
*  5)  MESSAGES NOT ACCEPTED ARE RETRANSMITTED UNTIL THEY ARE
*      FINALLY ACCEPTED.  THE DELAY BEFORE RETRANSMISSION IS EQUAL TO
*      THE LENGTH OF THE MESSAGE NOT ACCEPTED.
*  6)  THE PROBABILITY OF AN ERROR IN TRANSMISSION IS TAKEN AS
*      ONE CHARACTER IN 10,000.
*  7)  AFTER RECEIVING A MESSAGE, THAT NODE'S RECEIVER WILL BE BUSY
*      TO FURTHER MESSAGE RECEPTION FOR A FIXED TIME INTERVAL.
*
*  MESSAGE PARAMETER ASSIGNMENTS --
*
*  P(1)  DESTINATION NODE ADDRESS
*  P(2)  ORIGINATION NODE ADDRESS
*  P(3)  CURRENT NODE ADDRESS
*  P(4)  TOTAL MESSAGE LENGTH (DATA + HEADER)
*  P(5)  ACKNOWLEDGEMENT RESPONSE FIELD
*        0 -- NO RESPONSE
*        1 -- RECEIVER BUSY
*        2 -- ACCEPTED
*        3 -- TRANSMISSION ERROR
*  P(6)  ACTUAL CURRENT MESSAGE LENGTH (DATA OR ACK MSG)
*  P(7)  DELAY TIMER
*  P(8)  TRANSIT TIMER
*  P(9)  DELAY LOOP COUNTER
*
*
*  SAVEVALUE ASSIGNMENTS --
*
*  XH1  MEAN MESSAGE INTERARRIVAL TIME
*
*
*
*  VARIABLE ASSIGNMENTS --
```

```
*
NODE  VARIABLE   6          @NUMBER OF NODES IN NETWORK
DEST  BVARIABLE  P$P(1) GE P$P(2)    @INCR. DEST. ADDRS IF NEEDED
INCR  VARIABLE   (P$P(3)//V$NODE)+1      @INCREMENT CURRENT NODE
WAIT  VARIABLE   P$P(4)-R$S(V$V2)+1        @_TIME UNTIL DELAY STRGE AVAIL.
AMSG  VARIABLE   P$P(4)-6               @MSG. LENGTH - ACKLG. MSG.
RLAY  VARIABLE   P$P(6)-P$P(9)            @REMAINING DELAY TIME
BSYT  VARIABLE   5                    @RECEIVER BUSY TIME INTERVAL
M.I.A VARIABLE 600
M.LN  VARIABLE 50
V1    VARIABLE     P$P(1)
V2    VARIABLE     P$P(2)
V3    VARIABLE     P$P(3)
V4    VARIABLE     P$P(4)
V5    VARIABLE     P$P(5)
V6    VARIABLE     P$P(6)
V7    VARIABLE     P$P(7)
V8    VARIABLE     P$P(8)
V9    VARIABLE     P$P(9)
*
*
*  FUNCTION DEFINITIONS --
*
EXPON FUNCTION,EXP RF$2,1,1
*
UNIF FUNCTION,UNI RF$3,1,5
*
PRTY FUNCTION,UNI RF$4,0,7
*
*
*
*  DEFINE THE STORAGES, MATRICES, TABLES & QTABLES USED IN THE MODEL.
*
S(1)  CAPACITY     512
S(2)  CAPACITY     512
S(3)  CAPACITY     512
S(4)  CAPACITY     512
S(5)  CAPACITY     512
S(6)  CAPACITY     512
*
      MATRIX     RCVA(3,6)
      MATRIX     TRNA(3,6)
      MATRIX     TRAM(6,6)
*
GENAR TABLE IA,20.,20.,55
TRNAR TABLE IA,10.,10.,56
RTYAR TABLE IA,100.,100.,55
*
TRQTM TABLE MP$P(8),0.,20.,55
RCVTM TABLE MP$P(8),25.,25.,55
TTLTM TABLE M$1,30.,30.,55
ACKTM TABLE MP$P(8),10.,10.,55
TLATM TABLE M$1,30.,30.,55
DLYTM TABLE MP$P(7),0.,20.,56
*
MSGLN TABLE P$P(4),10.,10.,53
*
DBT1  TABLE     S$S(1),0.,10.,54
```

```
DBT2  TABLE     S$S(2).0..10..54
DBT3  TABLE     S$S(3).0..10..54
DBT4  TABLE     S$S(4).0..10..54
DBT5  TABLE     S$S(5).0..10..54
DBT6  TABLE     S$S(6).0..10..54
*
TQT1  QTABLE    0..25..55.Q(1)
TQT2  QTABLE    0..25..55.Q(2)
TQT3  QTABLE    0..25..55.Q(3)
TQT4  QTABLE    0..25..55.Q(4)
TQT5  QTABLE    0..25..55.Q(5)
TQT6  QTABLE    0..25..55.Q(6)
*
*
*
*  GENERATE MESSAGES FROM EACH OF 6 INDEPENDENT NODES.
*
MSG1  GENERATE 0  TIME(V$M.I.A*FN$EXPON)
      TRACE
      PRIORITY      FN$PRTY
      ASSIGN P(2).1
      ADVANCE GOTO(SETUP)
*
MSG2  GENERATE 0  TIME(V$M.I.A*FN$EXPON)
      PRIORITY      FN$PRTY
      ASSIGN P(2).2
      ADVANCE GOTO(SETUP)
*
MSG3  GENERATE 0  TIME(V$M.I.A*FN$EXPON)
      PRIORITY      FN$PRTY
      ASSIGN P(2).3
      ADVANCE GOTO(SETUP)
*
MSG4  GENERATE 0  TIME(V$M.I.A*FN$EXPON)
      PRIORITY      FN$PRTY
      ASSIGN P(2).4
      ADVANCE GOTO(SETUP)
*
MSG5  GENERATE 0  TIME(V$M.I.A*FN$EXPON)
      PRIORITY      FN$PRTY
      ASSIGN P(2).5
      ADVANCE GOTO(SETUP)
*
MSG6 GENERATE 0  TIME(V$M.I.A*FN$EXPON)
     PRIORITY      FN$PRTY
      ASSIGN P(2).6
      ADVANCE GOTO(SETUP)
*
*
*  SET UP DESTINATION NODE ADDRESS AND MESSAGE LENGTH.
*
SETUP ASSIGN P(1).FN$UNIF       @SET MSG DEST. ADDRS
      ASSIGN    P(1).V$V1+BV$DEST
      MSAVEX    TRAM(V$V2.V$V1).MX$TRAM(V$V2.V$V1)+1
      ASSIGN    P(4).V$M.LN*FN$EXPON
      ASSIGN    P(4).V$V4+9
      TABULATE   GENAR
      TABULATE   MSGLN
```

```
*
*
*  RETRANSMISSION ENTRY POINT FOR MESSAGES NOT ACCEPTED ORIGINALLY.
*
RETRY  ASSIGN P(3),V$V2
       ASSIGN P(6),V$V4
       TABULATE TRNAR
       MARK P(8)
*
*
*  WAIT FOR TRANSMITTER TO BE FREE & DELAY SPACE TO BE AVAILABLE.
*
      INQUEUE      Q(V$V2),PQ
FREE  GATE NU,F(V$V2)
      ADVANCE GOTO(+1,TRANS)
      COMPARE V$V4 GE R$S(V$V2)
      ADVANCE TIME(V$WAIT)
      ADVANCE GOTO(FREE)
*
*
*  OBTAIN CONTROL OF TRANSMITTER LONG ENOUGH TO SEND ONE MESSAGE.
*
TRANS SEIZE      F(V$V2)
      OUTQUEUE Q(V$V2),PQ
      TABULATE TRQTM
      MARK P(8)
      ADVANCE TIME(1)
      SPLIT 1,RECIV
*
      ADVANCE TIME(V$V4)
      RELEASE F(V$V2)
TRMSG MATCH AKMSG
      TERMINATE
*
*
*  CHECK IF MESSAGE IS ADDRESSED TO THIS NODE.  IF SO, REMOVE IT IF
*       THIS NODE IS NOT STILL BUSY FROM LAST MESSAGE RECEIVED AND SEND
*       THE PROPER ACKNOWLEDGEMENT MESSAGE IN REPLY.
*
RECIV ASSIGN     P(3),V$INCR
      ADVANCE    TIME(1)
      MARK       P(7)
      ADVANCE    GOTO(+1,ACKLG)
      COMPARE    V$V1 EQ V$V3
*
      ASSIGN     P(5),2
      ADVANCE GOTO(+1,RECVR)
      GATE LS,L(V$V1)
      ASSIGN  P(5),1
      ADVANCE GOTO(RECVD)
*
RECVR LOGIC S,L(V$V1)
RECVR1 ADVANCE GOTO(10:+4,+1)
      ADVANCE GOTO(10:+3,+1)
      ASSIGN   P(5),3
      ADVANCE  GOTO(RECVD)
      LOOP     P(6),RECVR1
*
```

```
RECVD ADVANCE TIME(V$AMSG)
      TABULATE  RCVTM
      TABULATE  TTLTM
      MARK      P(8)
      ASSIGN    P(6),6
      PRIORITY  7
      SPLIT     1,ACKLG
*
      ADVANCE   TIME(6)
      ADVANCE   GOTO(+1,RCTRM)
      COMPARE   V$V5 NE 1
      ADVANCE   TIME(V$BSYT)
      LOGIC R.L(V$V1)
*
RCTRM MSAVEX    RCVA(V$V5,V$V1),MX$RCVA(V$V5,V$V1)+1
      TERMINATE
*
*
*  CHECK IF ACKNOWLEDGEMENT MESSAGE IS ADDRESSED TO THIS NODE.  IF SO,
*      CHECK RESPONSE, UPDATE STATISTICS & RETRANSMIT IF NECESSARY.
*
ACKLG ADVANCE GOTO(+1,DELAY)
      COMPARE   V$V2 EQ V$V3
      MSAVEX    TRNA(V$V5,V$V2),MX$TRNA(V$V5,V$V2)+1
      ADVANCE TIME(6)
      TABULATE ACKTM
      TABULATE TLATM
AKMSG  MATCH   TRMSG
      ADVANCE GOTO(+1,ACKLD)
      COMPARE  V$V5 EQ 2
      TERMINATE,R  1
*
ACKLD TABULATE   RTYAR
      ADVANCE    GOTO(+1,+3)
      COMPARE    V$BSYT NE 0
      ADVANCE    TIME(V$BSYT)
      MARK
      ASSIGN     P(5),0
      ADVANCE GOTO(RETRY)
*
*
*  MESSAGE MUST BE RELAYED TO NEXT NODE, BUT MAY BE DELAYED HERE UNTIL
*      TRANSMITTER IS NO LONGER BUSY WITH LOCALLY GENERATED MESSAGES.
*
DELAY ASSIGN    P(9),V$V6
DELAY1 ADVANCE GOTO(+1,DRLAY)
      GATE U,F(V$V3)
      ENTER     S(V$V3),1
      ADVANCE   TIME(1)
      LOOP   P(9),DELAY1
*
DRLAY SEIZE     F(V$V3)
      TABULATE  DLYTM
      ADVANCE  TIME(1)
      SPLIT    1,RECIV
       ADVANCE  GOTO(+1,RELAY)
      COMPARE   V$V9 NE 0
      ADVANCE   TIME(V$V9)
      ADVANCE   GOTO(+1,RELS)
      COMPARE   V$V6 NE V$V9
RELAY ASSIGN    P(9),V$RLAY
RELAY1 LEAVE     S(V$V3),1
      ADVANCE   TIME(1)
      LOOP   P(9),RELAY1
*
RELS  RELEASE   F(V$V3)
      TERMINATE
*
*
*  RECORD CONTENTS OF EACH DELAY BUFFER AT REGULAR INTERVALS.
*
      GENERATE 0     TIME(20)
      TABULATE DBT1
      TABULATE DBT2
      TABULATE DBT3
      TABULATE DBT4
      TABULATE DBT5
      TABULATE DBT6
      TERMINATE
*
*  CONTROL CARDS FOR SIMULATION.
*
      START,NP 100
      RESET
      START 1000
      CLEAR
*
M.I.A  VARIABLE 420
      START,NP 100
      RESET
      START 1000
      CLEAR
*
M.I.A  VARIABLE 300
      START,NP 100
      RESET
      START 1000
      END
```

```
PLTHRU/CURRENT
  JOB
  ORDER.P  7  @ PARAMETERS
  ORDER.L  6.WAIT  @ LOGIC SWITCHES
  ORDER.UC  6      @USER CHAIN
  ORDER.F  12  @ FACILITIES
  ORDER.X  6.COUNT  @SAVEX VALUES FOR NODE STATES
  ORDER.Q  6  @QUEUE MESSAGES WAITING FOR GO
*
*
**
***    SIMULATION OF A 6 NODE RING PROCESSOR NETWORK
***    ALLOWING THE CONCURRENT GENERATION AND
***    TRANSMISSION OF ARBITRARY LENGTH MESSAGES THROUGH
***    THE USE OF A 'PLAYTHROUGH' PROTOCOL.
**
*
*
*  ASSUMPTIONS:
*   1) MESSAGES ARE GENERATED IN EACH NODE BY INDEPENDENT POISSON
*      PROCESSES. EACH HAVING THE SAME MEAN ARRIVAL RATE.
*   2) MESSAGE LENGTHS ARE ALSO EXPONENTIALLY DISTRIBUTED.
*      WITH A MEAN LENGTH OF 50 CHARACTERS AND A MAXIMUM
*      LENGTH OF 500 CHARACTERS
*   3) MESSAGES ALL HAVE AN ADDITIONAL 10 CHARACTERS OF
*      HEADER/TRAILER INFORMATION.
*   4) MESSAGES ENCOUNTERING CONTENTION FOR THE RING ARE
*      DELAYED UNTIL A PATH BECOMES AVAILABLE.
*
*
*
*
*       MESSAGE PARAMETER ASSIGNMENTS
*
*
*   P(1) DESTINATION NODE ADDRESS
*   P(2) ORIGIN (SOURCE) NODE ADDRESS
*   P(3) CURRENT NODE ADDRESS
*   P(4) TOTAL MESSAGE LENGTH (DATA + HEADER + TRAILER)
*   P(5) RANGE FIELD OF CURRENT NODE
*   P(6) MESSAGE QUEUEING TIMER
*   P(7) MESSAGE DISTANCE
**
*
*
*        SAVEX VALUES REPRESENTING NODE STATUS
*
*
*        X(1) THROUGH X(6)
*
*
*   0=FREE
*   1=AWAITING ACKNOWLEDGE
*   2=SOURCE
*   3=BRIDGE
*
*
*      *** VARIABLE ASSIGNMENTS ***
```

```
*
M.LEN       VARIABLE      50                    @MSG LENGTH MULTIPLIER
DEST        BVARIABLE     VSV1 GE VSV2
NODE        VARIABLE      6                      @# OF NODES IN NETWORK
INCR        VARIABLE      (P$P(3)//V$NODE)+1    @INCREMENT CURRENT NODE
M.I.A       VARIABLE      420                    @ MSG I.A.RATE
V1          VARIABLE      P$P(1)
V2          VARIABLE      P$P(2)
V3          VARIABLE      P$P(3)
V4          VARIABLE      P$P(4)
V5          VARIABLE      P$P(5)
V6        VARIABLE      P$P(6)
V7          VARIABLE      P$P(7)
*
*      *** FUNCTION DEFINITIONS ***
*
EXPON FUNCTION.C  RF$2.@  DEFINE EXPONENTIAL FUNCTION
+.0..0 .05..05129 .1..10536 .15..16252 .2..22314
+.25..28768 .3..35667 .35..43078 .4..51083 .45..59784
+.5..69315 .55..79851 .575..85567 .6..91629 .625..98083
+.65.1.04982 .675.1.12393 .7.1.20397 .725.1.29098 .75.1.38629
+.775.1.49165 .8.1.60944 .82.1.71480 .84.1.83258 .86.1.96611
+.88.2.12025 .9.2.30259 .91.2.40795 .92.2.52573 .93.2.65926
+.935.2.73337 .94.2.81341 .945.2.90042 .95.2.99573 .955.3.10109
+.96.3.21868 .965.3.35241 .97.3.50656 .974.3.64966 .977.3.77226
+.98.3.91202 .982.4.01738 .984.4.13517 .986.4.26870 .988.4.42285
+.99.4.60517 .991.4.71053 .992.4.82831 .993.4.96185 .994.5.11600
+.995.5.29832 .996.5.52146 .997.5.80914 .998.6.21461 .999.6.90776
+.9995.7.601 .9998.8.52 .9999.9.21 .99995.9.9 1.0.10.0
UNIF        FUNCTION.UNI    RF$3.1.6
*
*
*         *** MATRIX DEFINITIONS ***
*
            MATRIX        SRD(3.6)        @DEFINE SRC.RNGE.&DST. FOR EACH NOD
*
*
*          *** TABLES ***
*
*
TRNAR       TABLE         IA.10..10..56        @TOTAL XMITTED MSG IA RATE
MSGLN       TABLE         P$P(4).10..10..56    @TOTAL GENERATED MSG LENGTH
MSGAR       TABLE         IA.20..20..56        @GENERATED MSG IA RATE
CNLTM       TABLE        MP$P(1).5..15..56     @CONTROL PASSING TIME
TTLTM    TABLE       M$1.200..200..56          @TOTAL MESSAGE TRANSIT TIME
TLQTM    TABLE       M$1.0..200..56            @TOTAL QUEUEING TIME
TWO1     QTABLE      0..200..56.Q(1)           @QUEUE TIME AT NODE 1
TWO2     QTABLE      0..200..56.Q(2)           @QUEUE TIME AT NODE 2
TWO3     QTABLE      0..200..56.Q(3)           @QUEUE TIME AT NODE 3
TWO4     QTABLE      0..200..56.Q(4)           @QUEUE TIME AT NODE 4
TWO5     QTABLE      0..200..56.Q(5)           @QUEUE TIME AT NODE 5
TWO6     QTABLE      0..200..56.Q(6)           @QUEUE TIME AT NODE 6
TLQ1     TABLE       M$1.0..100..200           @QUEUE TIME FOR 1 NODE DX.
TLQ2     TABLE       M$1.0..100..200             @2 NODES TRAVELLED
TLQ3     TABLE       M$1.0..100..200             @3 NODES TRAVELLED
TLQ4     TABLE       M$1.0..100..200             @4 NODES TRAVELLED
TLQ5     TABLE       M$1.0..100..200             @5 NODES TRAVELLED
*
```

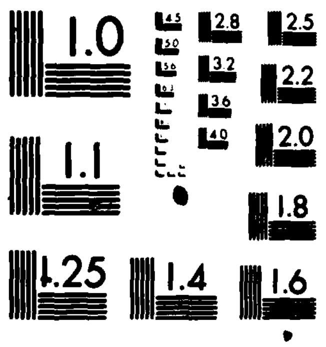
```
*
        INITIAL        SRD(2.1-6).1.2.3.4.5.6
        INITIAL        L(1-6).R
        INITIAL        WAIT.S
*
*
*
*    GENERATE THE 'GO' MESSAGE
*
            GENERATE        0.1.5          @CREATE ONE COPY OF GO
            ASSIGN          P(3).FNSUNIF  @PICK START NODE RANDOMLY
GOMSG   MARK           P(1)                @MARK THE GO TRANSIT TIMER
        ADVANCE         GO TO(+1.GOMSG.1)
        COMPARE        XSX(VSV3) EQ 0          @SEE IF NODE IS FREE
        ADVANCE        GOTO(+1.GOMSG.1)
        COMPARE        CHSUC(VSV3) NE 0    @SEE IF USER CHAIN IS EMPTY
        LOGIC          R.WAIT              @ WAIT FOR CTL MSG TO LEAVE
        SAVEX          COUNT.CHSUC(VSV3)-1        @INITIALIZE CHAIN COUNTER
        UNLINK         UC(VSV3).1.SETUP.3         @UNLINK A MESSAGE
        GATE           LS.WAIT                     @WAIT FOR  CTL MSG TO LV
GOMSG.1     LOGIC           S.L(VSV3)              @GATE OUT THE STOP MSG
        PREEMPT.PRI   F(VSV3)    TIME(1)           @XMIT GO TO NXT NODE
        RETURN.PRI     F(VSV3)
        LOGIC          R.L(VSV3)                   @ONLY ONE STOP AT A TIME
        TABULATE       CNLTM                       @ TABULATE CTL PASSING TIME
            ASSIGN          P(3).VSINCR     @GO TO NEXT NODE
            ADVANCE         GO TO(GOMSG)    @MOVE ON TO NEXT NODE
*
*     INITIATE MESSAGESFROM EACH NODE EXPONENTIALLY
*
MSG1        GENERATE        0.0.0   TIME(VSM.I.A*FNSEXPON)@  CREATE A MSG
            ASSIGN          P(2).1          @ORIGIN ADDRESS = 1
            ADVANCE         GO TO(SETUP)    @ GO SET OTHER PARAMS.
MSG2        GENERATE        0.0.0   TIME(VSM.I.A*FNSEXPON)
            ASSIGN          P(2).2
            ADVANCE         GO TO(SETUP)
MSG3        GENERATE        0.0.0   TIME(VSM.I.A*FNSEXPON)
            ASSIGN          P(2).3
            ADVANCE         GO TO(SETUP)
MSG4        GENERATE        0.0.0   TIME(VSM.I.A*FNSEXPON)
            ASSIGN          P(2).4
            ADVANCE         GO TO(SETUP)
MSG5        GENERATE        0.0.0   TIME(VSM.I.A*FNSEXPON)
            ASSIGN          P(2).5
            ADVANCE         GO TO(SETUP)
MSG6        GENERATE        0.0.0   TIME(VSM.I.A*FNSEXPON)
            ASSIGN          P(2).6
            ADVANCE         GO TO(SETUP)
*
*
*    NOW SET UP OTHER MESSAGE PARAMETERS.
*
*
*
*
*SET OTHER MESSAGE PARAMETERS
*
*
```

```
SETUP    ASSIGN      P(1).FN$UNIF             @SET DEST ADDRESS
         ASSIGN      P(1).P$P(1)+BV$DEST
SETUP.Z  ADVANCE     GOTO(+1.SETUP.A)
         COMPARE     V$V1 GT V$V2             @SEE IF DEST > SOURCE
         ASSIGN      P(7).(V$V1-V$V2)         @IF SOSET UP DISTANCE FIELD
         ADVANCE     GOTO(SETUP.1)            @GO CONTINUE
SETUP.A  ASSIGN      P(7).(V$V2-V$V1)         @GET COMPLEMENT OF DISTANCE
         ASSIGN      P(7).6-V$V7              @AND COMPLEMENT IT
SETUP.1  ASSIGN      P(3).P$P(2)              @SET UP CURRENT NODE AT ORIGIN
         ASSIGN      P(4).V$M.LEN+FN$EXPON    @SET MSG LENGTH
         ASSIGN      P(4).P$P(4)+10            @ADD HEADER/GO/TRAILER
         PRIORITY    10                        @SET PRIORITY
         ASSIGN      P(5).MX$SRD(2.V$V2)      @SET RANGE PARAM TO RNG OF ORIG
         TABULATE    MSGAR                    @TABULATE MESSAGE IA TIME
         TABULATE    MSGLN                    @TABULATE MESSAGE LENGTH
         INQUEUE     Q(V$V3).P(6)             @ WAIT FOR GO
         LINK.U      UC(V$V3).FIFO            @ON THE USER CHAIN
*
*
*     CHECK THAT THE DEST IS WITHIN THE
*     RANGE OF THE ORIGIN
*     THIS IS SO UNDER THE FOLLOWING CONDITIONS:
*     IF SOURCE < DEST. RNGE OF SOURCE MUST BE <= SOURCE OR >= DEST.
*     IF SOURCE > DEST. RNGE OF SOURCE MUST BE <= SOURCE AND >= DEST.
*
*
*
SETUP.3      ADVANCE        GO TO(+1.SETUP.4)
             COMPARE        MX$SRD(2.V$V3) LE P$P(2) @SEE IF RANGE IS LE ORIGIN
             ADVANCE        GO TO(+1.SETUP.OK)
             COMPARE        P$P(1) LT P$P(2)
             ADVANCE        GO TO(SETUP.5)
SETUP.4      ADVANCE        GO TO(+1.SETUP.5)
             COMPARE        P$P(1) LT P$P(2)
             ADVANCE        GO TO (SETUP.NG)
SETUP.5      ADVANCE        GO TO(+1.SETUP.NG)
             COMPARE        MX$SRD(2.V$V3) GE P$P(1)
             ADVANCE        GO TO(SETUP.OK)
*
*
*     ANY TRANSACTION REACHING THE FOLLOWING BLOCK
*     CANNOT BE SENT YET DUE TO INADEQUATE RANGE
*     OF ITS TRANSMITTER.  IT IS RELINKED ONTO THE
*    USER CHAIN AND WILL BE RETRIED ON A SUBSEQUENT
*    PASSING OF GO.
*
*
SETUP.NG ADVANCE     GOTO(+1.SETUP.N1)      @ SEE IF THIS WAS LAST ON UC
         COMPARE     X$COUNT EQ 0           @IF SO. JUST RELINK IT
         LOGIC       S.WAIT                 @LET GO CONTINUE
         LINK.U      UC(V$V3).FIFO          @RELINK
SETUP.N1 SAVEX     . COUNT.X$COUNT-1        @ELSE DECREMENT CHAIN COUNTER
         UNLINK      UC(V$V3).1.SETUP.3     @TRY THE NEXT ENTRY
         LINK.U      UC(V$V3).FIFO          @RELINK THIS ONE
*
*
*     TRANSACTIONS REACHING THE FOLLOWING BLOCK ARE
*     WITHIN THE RANGE OF THEIR ORIGINS. AND ARE SENT.
```

```
*
*
SETUP.OK OUTQUEUE       Q(V$V3).P(6)            @STOP QUEUEING
*
*
*     THE FOLLOWING CODE DETERMINES IN WHICH TABLE TO
*     TO TABULATE QUEUEING TIME. DEPENDENT UPON
*     MESSAGE LENGTH
*
*
         ADVANCE        GOTO(+1.TEST2)
         COMPARE        V$V7 EQ 1
         ADVANCE        GO TO(Q1)
TEST2    ADVANCE        GOTO(+1.TEST3)
         COMPARE        V$V7 EQ 2
         ADVANCE        GOTO(Q2)
TEST3    ADVANCE        GOTO(+1.TEST4)
         COMPARE        V$V7 EQ 3
         ADVANCE        GOTO(Q3)
TEST4    ADVANCE        GOTO(+1.Q5)
         COMPARE        V$V7 EQ 4
         ADVANCE        GOTO(Q4)
Q1       TABULATE       TLQ1
         ADVANCE        GOTO(SETUP.B)
Q2       TABULATE       TLQ2
         ADVANCE        GOTO(SETUP.B)
Q3       TABULATE       TLQ3
         ADVANCE        GOTO(SETUP.B)
Q4       TABULATE       TLQ4
         ADVANCE        GOTO(SETUP.B)
Q5       TABULATE       TLQ5
SETUP.B  TABULATE       TLQTM                   @TABULATE TOTAL QUEUE TIME
         TABULATE       TRNAR                    @NOTE IA RATE OF SUCCESSES
         ADVANCE        GOTO(+1.SETUP.O1)
         COMPARE        X$COUNT EQ CH$UC(V$V3)  @ SEE IF MSG WAS 1ST ON CHAIN
         SAVEX          COUNT.0                   @IF SO. RESET CHAIN COUNTER
         ADVANCE        GOTO(SETUP.O3)            @AND GO XMIT IT
SETUP.O1 ADVANCE        GOTO(+1.SETUP.O3)         @SEE IF MSG WAS LAST ON CHAIN
         COMPARE        X$COUNT NE 0
         UNLINK         UC(V$V3).1.SETUP.O2       @IF NOT. SHIFT CHAIN AROUND
         SAVEX          COUNT.X$COUNT-1           @SO THAT MESSAGE ORDER
         ADVANCE        GOTO(SETUP.O1)            @IS NOT CHANGED
SETUP.O2 LINK.U         UC(V$V3).FIFO              @RELINK HERE
*
*
*
*
*
SETUP.O3 SAVEX          X(V$V2).2                  @SET UP AS A SOURCE
         PREEMPT.PRI    F(V$V2) TIME(3)            @START IS BEFORE GO
         RETURN.PRI     F(V$V2)
         SPLIT          1.BEGIN                 @OTHER ONE GOES ON TO NEXT NODE
         PRIORITY       1                       @THIS ONE IS DATA MSG AT ORGIN
         LOGIC          S.WAIT                  @LET GO CONTINUE
         PREEMPT.PRI    F(V$V2) TIME(V$V4-6)  @HOLD ORIGIN FOR MESSAGE LENGTH
         RETURN.PRI     F(V$V2)
         PRIORITY       10                        @NOW IT BECOMES STOP OR TAIL
         GATE           LS.L(V$V2)                    @AFTER THE MESSAGE.
         PREEMPT.PRI    F(V$V2) TIME(3)          @STOP OR TAIL MUST
         RETURN.PRI     F(V$V2)                  @WAIT FOR GO.
         SAVEX          X(V$V2).1               @PLACE IN AWAIT ACK. STATE
         ADVANCE        GO TO(END1)             @GO PROCESS STOP OR TAIL
*
*
*     SEE IF THERE ARE ANY OTHER MESSAGES ON THE
*     LOOP. IF NOT. GO TO START. IF SO. GO TO HEAD.
*
*
BEGIN    ADVANCE        GOTO(+1.HEAD)
         COMPARE        MX$SRD(2.V$V2) EQ P$P(2)
         ADVANCE        GO TO(START1)
*
*
*     SEND THE MESSAGE. ALTERING SOURCE. RANGE. AND
*     DESTINATION REGISTERS ALONG THE WAY.  SET ALL
*     INTERMEDIATE NODES TO THE BRIDGE STATE.
*
*
START1    MSAVEX         SRD(1.V$V3).0            @SET SOURCE
         MSAVEX         SRD(3.V$V3).V$V1         @SET DEST
START1.1 ASSIGN         P(3).V$INCR              @GO TO NEXT NODE
         ADVANCE        GO TO(+1.SET)            @SEE IF DEST YET
         COMPAPE        P$P(3) NE P$P(1)         @IF NOT DEST YET.
         MSAVEX         SRD(1.V$V3).V$V2         @SET SOURCE.
         MSAVEX         SRD(2.V$V3).V$V2         @RANGE.
         MSAVEX         SRD(3.V$V3).V$V1         @AND DEST REGISTERS
         SAVEX          X(V$V3).3                @SET UP A BRIDGE
         PREEMPT.PRI    F(V$V3) TIME(3)          @TRANSMIT THE MESSAGE
         RETURN.PRI     F(V$V3)
         SPLIT          1.START1.2               @GO HOLD THE NODE FOR MSGLN
         ADVANCE        GOTO(START1.1)              @CONTINUE
*
*
*     HOLD EACH BRIDGE FOR THE LENGTH OF THE MESSAGE
*
*
START1.2 PRIORITY       1                          @DATA PRIORITY
         PREEMPT.PRI    F(V$V3)  TIME(V$V4-6)      @XMIT IT
         RETURN.PRI     F(V$V3)
         TERMINATE                                   @AT THIS NODE ONLY
*
*
*     AT THE DEST. START IS TRANSFORMED INTO SET
*
*
SET      MSAVEX         SRD(1.V$V3).V$V2         @SET SOURCE.
         MSAVEX         SRD(2.V$V3).V$V2         @RANGE.
         MSAVEX         SRD(3.V$V3).0            @AND DEST.
*
*
*     SET RESETS RANGE REGISTERS UNTIL REACHING THE ORIGIN
*
*
SET.1    PREEMPT.PRI    F(V$V3) TIME(3)          @TRANSMIT THE CONTROL MSG
         RETURN.PRI     F(V$V3)
         ASSIGN         P(3).V$INCR              @GO TO NEXT NODE
```

```
               ADVANCE        GO TO(+1.SETFIN)
               COMPARE        P$P(3) NE P$P(2)          @IF NOT DONE YET.
               MSAVEX         SRD(2.V$V3).V$V2          @SET RANGE
               ADVANCE        GO TO(SET.1)              @AND DO NEXT NODE
*
*
*     AT SETFIN. SET HAS COMPLETED ITS TASK. AND IS TERMINATED
*
*
SETFIN         TERMINATE
*
*
*     AT END. THE DETERMINATION IS MADE AS TO
*     WHETHER TO USE STOP OR TAIL.
*
*
END1            ADVANCE        GO TO(+1.TAIL)
               COMPARE        MX$SRD(2.V$V2) EQ V$V2
               ADVANCE        GO TO(STOP1)
*
*
*     STOP MUST UNDO THE ACTION OF START
*
*
STOP1           ASSIGN         P(3).V$INCR               @GO TO NEXT NODE
               MSAVEX         SRD(1.V$V3).0             @RESET SOURCE
               MSAVEX         SRD(2.V$V3).V$V3          @RANGE.
               MSAVEX         SRD(3.V$V3).0             @AND DEST.
               ADVANCE        GO TO(+1.REESET)           @IF AT DEST. GO TO RESET
               COMPARE        P$P(3) NE P$P(1)
               SAVEX          X(V$V3).0                 @SET AS FREE STATE
               PREEMPT.PRI    F(V$V3)   TIME(3)            @XMIT THE CTL MSG
               RETURN.PRI     F(V$V3)
               ADVANCE        GO TO(STOP1)               @CONTINUE
*
*
*     AT THE DEST. STOP BECOMES RESET
*
*
REESET         PREEMPT.PRI    F(V$V3) TIME(3)           @TRANSMIT THE MESSAGE
               RETURN.PRI     F(V$V3)
               ASSIGN         P(3).V$INCR
               ADVANCE        GO TO(+1.RESFIN)          @SEE IF AT ORIGIN
               COMPARE        P$P(3) NE P$P(2)          @IF NOT DONE YET.
               MSAVEX         SRD(2.V$V3).V$V3          @RESET RANGE
               ADVANCE        GO TO(REESET)
*
*
*     AT THIS POINT. RESET HAS REACHED THE ORIGIN.  IT WILL
*     RESET IT. AND REMOVE A TRANSACTION FROM THE SYSTEM.
*
*
RESFIN         MSAVEX         SRD(1.V$V3).0             @RESET SOURCE.
               MSAVEX         SRD(2.V$V3).V$V3          @RANGE.
               MSAVEX         SRD(3.V$V3).0             @AND DEST.
               SAVEX          X(V$V3).0                 @SET TO FREE STATE
               TABULATE       TTLTM                      @TABULATE TOTAL XMIT TIME
               TERMINATE.R    1                          @REMOVE IT
```

115012



```
*
*
*      'HEAD' CONTROL MESSAGES ENTER AT THIS POINT
*
*
HEAD     MSAVEX          SRD(3.V$V3).V$V1          @SET DEST. REG. OF ORIGIN
HEAD.1   ASSIGN          P(3).V$INCR
         ADVANCE         GO TO(+1.INIT)
         COMPARE         P$P(3) NE P$P(1)          @IF NOT DEST YET.
         MSAVEX          SRD(1.V$V3).V$V2          @SET SOURCE
         MSAVEX          SRD(3.V$V3).V$V1          @AND DEST
         SAVEX           X(V$V3).3                 @STATE IS A BRIDGE
         PREEMPT.PRI     F(V$V3) TIME(3)           @TRANSMIT CONTROL MSG
         RETURN.PRI      F(V$V3)
         SPLIT           1.HEAD.2                   @HOLD THE BRIDGE FOR MSGLN
         ADVANCE         GO TO(HEAD.1)
*
*
*      HOLD EACH NODE FOR THE LENGTH OF THE MESSAGE
*
*
HEAD.2   PRIORITY        1                          @DATA MESSAGE PRIORITY
         PREEMPT.PRI     F(V$V3)   TIME(V$V4-8)      @HOLD THE BRIDGES
         RETURN.PRI      F(V$V3)                     @WHEN FINISHED.
         TERMINATE                                   @RETURN IT
*
*
*      AT INIT. THE AFFECTED REGION IS SEARCHED OUT
*
*
INIT     MSAVEX          SRD(1.V$V3).V$V2          @SET SOURCE
INIT.1   ADVANCE         GO TO(+1.INIT.2)
         COMPARE         X$X(V$V3) EQ 2            @IS THIS A SOURCE
         ADVANCE         GO TO(+1.INITX)           @IF SO. SEE IF RANGE IS DIFFERE
         COMPARE         MX$SRD(2.V$V3) NE MX$SRD(2.V$V2)   @FROM NEW ORIGIN
INIT.2    PREEMPT.PRI     F(V$V3) TIME(3)           @XMIT THE CNTRL MSG
         RETURN.PRI      F(V$V3)
         ASSIGN          P(3).V$INCR               @IF DIFFERENT. KEEP GOING
         ADVANCE         GO TO(INIT.1)             @CONTINUE
*
*
*      NOW CHANGE INIT TO INIT*
*
*
INITX    MSAVEX          SRD(2.V$V3).V$V2          @RESET THE RANGE
         PREEMPT.PRI     F(V$V3) TIME(3)             @XMIT THE CTL MSG
         RETURN.PRI      F(V$V3)
         ASSIGN          P(3).V$INCR
         ADVANCE         GO TO(+1.INITX.1)
         COMPARE         V$V3 NE V$V2              @IF AT NEW ORIGIN. THEN DONE
         ADVANCE         GO TO(INITX)              @ELSE DO NEXT NODE
*
*
*      INIT* HAS COMPLETED ITS TASK
*
*
INITX.1  TERMINATE
*
```

```
*
*
*     THE WORK OF TAIL IS NOW TO BE STARTED.(MSG ENDING
*     ON A MULTIPLE MESSAGE LOOP.
*
*
TAIL     MSAVEX          SRD(3.V$V3).0              @DEST BECOMES ZERO.
TAIL.1   ASSIGN          P(3).V$INCR
         ADVANCE         GO TO(+1,TERM)
         COMPARE         V$V3 NE V$V1               @SEE IF DEST YET
         MSAVEX          SRD(1.V$V3).0              @RESET SOURCE REG.
         MSAVEX          SRD(3.V$V3).0                @AND DEST REGISTER
         SAVEX           X(V$V3).0                  @REVERTS TO FREE STATE
         PREEMPT.PRI     F(V$V3) TIME(3)              @XMIT THE CTL MSG
         RETURN.PRI      F(V$V3)
         ADVANCE         GO TO(TAIL.1)                @AND CONTINUE
*
*
*     NOW AT DESTINATION.  CONVERT TAIL TO TERM
*     AND SEARCH FOR AFFECTED REGION.
*
*
TERM     MSAVEX          SRD(1,V$V3).0              @SOURCE DISAPPEARS
TERM.1   ADVANCE         GO TO(+1,TERM.2)
         COMPARE         X$X(V$V3) EQ 2             @SEE IF A SOURCE
         ADVANCE         GO TO(+1,TERMX)            @IF SO. IS RANGE
         COMPARE         MX$SRD(2,V$V3) NE V$V2     @THE NEW ORIGIN
TERM.2   PREEMPT.PRI     F(V$V3) TIME(3)            @TRANSMIT THE CNTRL MSG
         RETURN.PRI      F(V$V3)
         ASSIGN          P(3).V$INCR                @IF NOT. GO TO NEXT NODE
         ADVANCE         GO TO(TERM.1)              @CONTINUE
*
*
*     NOW CONVERT TERM TO TERM* AND RESET THE AFFECTED REGION
*
*
TERMX    MSAVEX          SRD(2.V$V3).MX$SRD(2.V$V2) @RESET RANGE
         PREEMPT.PRI     F(V$V3) TIME(3)              @XMIT THE CTL MSG
         RETURN.PRI      F(V$V3)
         ASSIGN          P(3).V$INCR
         ADVANCE         GO TO(+1,TERMX.1)
         COMPARE         V$V3 NE V$V2               @IF AT NEW ORIGIN. DONE
         ADVANCE         GO TO(TERMX)               @ELSE DO NEXT NODE
*
*
*     TERM* HAS COMPLETED ITS TASK
*
*
TERMX.1  SAVEX           X(V$V3).0                  @RESET TO FREE STATE
         TABULATE        TTLTM
         TERMINATE.R     1                          @REMOVE A TRANSACTION
      START.NP   100
        RESET
        START    1000
      CLEAR
M.I.A  VARIABLE 300
      START.NP 100
      RESET
```

```
522          * START 1000
523      M.I.A VARIABLE 600
524            START.NP   100
525            RESET
526            START 1000
527              END
```

APPENDIX D

GLOSSARY

Next page is blank.

APPENDIX D

GLOSSARY

AMSAA	-	US Army Materiel Systems Analysis Activity
APG	-	Aberdeen Proving Ground, Maryland
ARRADCOM	-	US Army Armament Research and Development Command
ASAS FSD	-	All Source Analysis System Full Scale Development
C^3A	-	Command, Control, and Communications Analysis
CDC	-	Trademark and abbreviation for the Control Data Corporation
CPU	-	Central Processing Unit of computer systems
CSD	-	Combat Support Division
DA	-	Department of the Army
DLCN	-	Distributed Loop Computer Network, The Ohio State University.
DLCNNE	-	Modified simulation model for DLCN with no errors in character transmission
GPSS	-	Either of two simulation language dialects called "General Purpose Simulation System" by IBM and called "General Purpose Systems Simulator" by UNIVAC
IBM	-	Trademark and abbreviation for International Business Machines Corporation
OPTADS	-	Operations Tactical Data Systems
PM	-	Program or Project Manager
SACDIN	-	Stragetic Air Command Digital Network
SIGMA	-	Name of force level maneuver control system
SIMSCRIPT	-	Generic name of a computer programming language developed at the RAND Corporation for discrete event simulation with a version marketed under the trademark SIMSCRIPT II.5 by Consolidated Analysis Centers, Inc.
TOS CASE	-	Tactical Operations Systems for Corps and Subordinate Echelons
UNIVAC	-	Trademark and name of the Sperry UNIVAC Division of the Sperry Rand Corporation

Next page is blank.

DISTRIBUTION LIST

No. of Copies	Organization
12	Commander Defense Technical Information Center ATTN: DDC-TC Cameron Station Alexandria, VA 22314
5	Commander US Army Materiel Development & Readiness Command ATTN: DRCPA-P DRCCE DRCDE-SB DRCQA DRCMS 5001 Eisenhower Avenue Alexandria, VA 22333
1	Commander US Army Communications Command Fort Huachuca, AZ 85613
1	Commander US Army Communications Systems Agency ATTN: CCM Fort Monmouth, NJ 07703
7	Commander US Army Communications-Electronics Command ATTN: DRCPM-OTDS DRCPM-ATC DRDCO-SA DRDCO-TCS DRDCO-COM-RN DRDCO-SEI DRDCO-ST-SA (Mr. Sizelove) Fort Monmouth, NJ 07703
2	Commander US Army Electronic Proving Ground ATTN: STEEP-TC STEEP-MT-DD Fort Huachuca, AZ 85613
3	Commander US Army White Sands Missile Range ATTN: STEWS-TE-P STEWS-TE-MC STEWS-TE-A White Sands Missile Range, NM 88002
2	Commander US Army Electronic Research & Development Command ATTN: DRDEL-NW-EB (Mr. Gornak) DRDEL-AP-OA 2800 Powder Mill Road Adelphi, MD 20783
1	Commander US Army Missile Command ATTN: DRSMI-DM DRSMI-DS Redstone Arsenal, AL 35809
1	Director US Army Research Office PO Box 12211 Research Triangle Park, NC 27709
3	Director US Army TRADOC Systems Analysis Activity ATTN: ATAA-SL ATAA-T ATAA-TC (Mr. Mathiasen) White Sands Missile Range, NM 88002
1	Director Joint Tactical Communications (TRI-TAC) ATTN: TT-D Fort Monmouth, NJ 07703

DISTRIBUTION LIST (Continued)

No. of Copies	Organization
2	Director Army Tactical Data Systems Office, AMC Air Defense Control and Coordination Systems Directorate ATTN: DRCPM-TDS-RA-E DRCPM-TDS-RA-P Redstone Arsenal, AL 35809
2	Chief Defense Logistics Studies Information Exchange US Army Logistics Management Center ATTN: DRXMC-D Fort Lee, VA 23801
1	Commander US Army Concepts Analysis Agency 8120 Woodmont Avenue Bethesda, MD 20014
1	Office, Secretary of the Army Deputy Under Secretary ATTN: SAUS-OR (Dr. Fallin) Rm 2E614, The Pentagon Washington, DC 20310
1	Dr. Alan Mink National Bureau of Standards Technology A-214 Washington, DC 20234
1	Reliability Analysis Center ATTN: Mr. I. L. Krulac Griffiss AFB, NY 13441
	Aberdeen Proving Ground
3	Cdr, USATECOM ATTN: DRSTE DRSTE-CS-A DRSTE-AD-S (Mr. Whiting) Building 314
1	Director, BRL Building 328
1	Director, BRL ATTN: DRDAR-TSB-S (STINFO Br) Building 305
1	Director, HEL Building 520

ILMED

8